Static-Dynamic Disentanglement for Efficient Multi-Frame Vision-Language-Action Models

Weikang Qiu[*]
Yale University

Huashuo Lei[*]
The Hong Kong University of Science and Technology (Guangzhou)

Tinglin Huang
Yale University

Aosong Feng
Yale University

Rex Ying
Yale University

Abstract

Vision-Language-Action (VLA) models have recently emerged as a promising paradigm for generalist robotic control. Built upon vision–language model (VLM) architectures, VLAs predict actions conditioned on visual observations and language instructions, achieving strong performance and generalization across tasks. However, VLAs face two major challenges: a limited context window for input frames and inefficient inference due to the quadratic attention complexity and large parameter counts. To this end, we propose DySta, a framework that disentangles visual inputs into multi-level static and dynamic tokens, which enables (1) retaining a single copy of static tokens across frames to significantly reduce context length, and (2) reusing the key–value (KV) cache of static tokens through a lightweight recache gate that updates only when necessary. This design enables efficient multi-frame integration and efficient inference. In addition, we introduce a new benchmark that more effectively evaluates the multi-frame integration ability of VLAs. Experiments show that DySta improves multi-frame integration by 24.5% across metrics on our benchmark and 23.3% in absolute success rate on real-world memory-dependent tasks, while accelerating inference by 2.0× (with +2.3% success rate) on simulation benchmarks and 2.2× (with +10.6% success rate) on real-world general tasks.

1 Introduction

Vision-Language-Action (VLA) models [20, 18, 37, 19, 66, 63, 3, 12] have recently emerged as a powerful paradigm for generalist robotic control. Trained on large-scale heterogeneous datasets such as Open-X-Embodiment [54], which aggregate expert demonstrations across diverse robotic tasks, state-of-the-art VLA models exhibit impressive task performance. However, they are typically built on top of large vision–language models [24, 17, 2, 48], whose large parameter counts and quadratic-complexity contextual processing [53] translate into two central deployment bottlenecks: (1) a constrained context budget that limits how many past observations can be ingested, and (2) high per-step inference latency that hinders real-time control. These manifest as the two challenges of *multi-frame integration* and *efficient inference* elaborated below and illustrated in Figure 1.

Multi-frame Integration. Most current VLAs operate in a memoryless manner [18, 20, 37], taking only the current observation as input. As a result, they struggle with tasks that require temporal dependency or memory tracking. For instance, when instructed to press a button, a VLA must remember whether the button has already been pressed; otherwise, it may repeat the same action

[*]Equal contribution.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

indefinitely. A straightforward solution is to include previous observations in the model input. However, VLA's vision backbones typically produce hundreds of visual tokens per image, leading to prohibitively long contexts when multiple frames are concatenated for the transformer-based architecture with quadratic complexity in sequence length. Existing methods often rely on indirect or compressed representations [15, 51], or expose multiple frames to the decoder head only [45], which either risks significant information loss or bypasses the language model's ability to jointly reason over multiple frames.

Efficient Inference. Due to their large model sizes, VLAs incur long latency for each forward pass. In real-world settings, however, robotic systems are often expected to respond promptly—for example, a household assistant should complete tasks as quickly as possible, while safety-critical scenarios such as spill containment or fire response may require near-instantaneous reactions. Moreover, recent post-training approaches for VLAs [47, 27], rely on reinforcement learning [49, 42, 44] and require extensive rollouts during training, making inference speed a key bottleneck. While existing work has explored improving VLA efficiency through generic techniques for general machine learning models, such as quantization, token pruning, or layer pruning [57, 60, 59, 18], these approaches do not leverage the intrinsic characteristics of VLA tasks. Recent works [57, 23, 50] exploit temporal redundancy by reusing computations across consecutive frames, via KV-cache reuse or action reuse. Nonetheless, these techniques rely on heuristic, non-learnable criteria and often assume that visual similarity in pixel space indicates the temporal-consistency of their latent representations—an assumption that is invalid in transformer-based vision and language backbones, as illustrated in Figure 2.

To address these challenges, we propose DySta. We focus on a broad and practically important class of robotic scenarios—such as tabletop manipulation, household chores, and warehouse picking—where the agent operates within relatively stable scenes or undergoes discrete, infrequent scene transitions. In this regime, much of the visual information in a scene remains static or changes slowly over time: the background, objects remaining still, and even moved objects whose visual appearances stay largely invariant. Building on this observation, we explicitly disentangle visual tokens into dynamic tokens and multi-level static tokens with different temporal persistence. This design yields two key benefits. First, instead of conditioning on a single image as in prior work, our model can ingest multi-step observations while maintaining a compact context: static tokens are included only once in the input sequence across timesteps, while only dynamic tokens from multiple steps are concatenated over time. This substantially reduces the effective context length and enables multi-frame integration. Second, our method improves the inference efficiency by reusing key–value (KV) caches associated with static tokens from previous steps. We also introduce a recache gate module that determines whether previously cached static tokens should be reused or recomputed, which further improves performance while minimizing the inference latency.

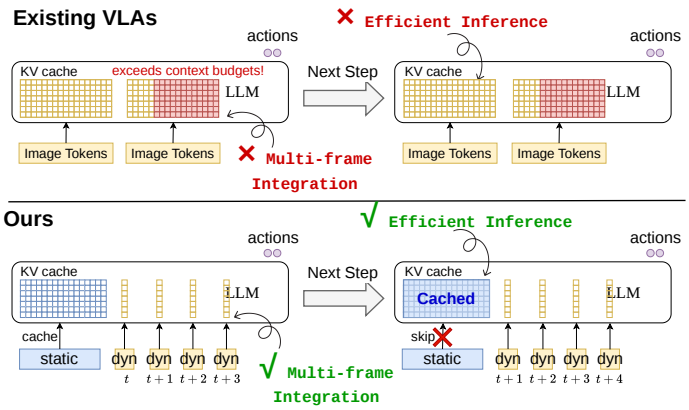


Figure 1: Our model solves two main challenges in existing VLAs with the proposed static-dynamic disentanglement. 1) By keeping one copy of static tokens across all timesteps, our model is able to squash observations of multiple steps to the model's context; 2) By moving static tokens in front of all dynamic tokens, our model could reuse the KV-cache of previous timesteps during rollouts.

In addition, we observe that existing simulation benchmarks largely fail to assess a VLA's multi-frame integration capabilities. For example, temporally dependent tasks such as placing objects into a basket do not require remembering past trajectories. To address this gap, we design a new simulation benchmark inspired by principles of human episodic memory, which provides a more effective evaluation of temporal reasoning and memory usage than prior benchmarks [22, 21].

To evaluate multi-frame integration, we assess our model on the proposed benchmark. Our method achieves a 24.5% improvement across all metrics and 23.3% in absolute success rate on real-world

experiments. To evaluate inference efficiency, we benchmark our approach on public simulation benchmarks [22, 21] and real-world experiments. On SimplerEnv [21], DySta improves the success rate by 3.9% over the strongest baseline while achieving a $2.26\times$ inference speedup. In real-world experiments, DySta improves the average success rate by 10.6% over the base model and achieves a $2.21\times$ speedup. These results demonstrate that DySta not only enhances multi-frame integration, but also substantially reduces inference latency. In summary, our contributions are as follows:

- We propose DySta, which enables multi-frame integration and efficient inference by disentangling image tokens to dynamic tokens and multi-level static tokens with different temporal persistence.
- We introduce a trainable recache gate that adaptively determines when to refresh the cache or reuse previously cached representations, further improving the performance while minimizing the inference latency.
- We introduce LIBERO-Memory, a new benchmark that more effectively evaluates a VLA's multi-frame integration ability by temporally dependent tasks.

2 Related Works

Vision-Language-Action Models Vision-language models (VLMs) [24, 56] have demonstrated strong performance in image-related and cross-modal tasks. Leveraging these capabilities, vision-language-action (VLA) models further finetune VLMs on large-scale robotic datasets [54, 22], which usually consist of extensive human demonstrations. At each timestep, VLAs [3, 12, 18, 20, 37] usually take the current environment observation (image) and a language instruction as inputs, and are trained to output actions of the next step. Although they have demonstrated high performance on certain benchmarks, due to the large parameters, they are 1) inefficient in rollout and 2) unable to incorporate historical frames in the context. In contrast, our model, by disentangling the static and dynamic components, is efficient in inference by leveraging the KV-cache of static components, and is able to incorporate a long history for temporally dependent tasks.

VLA Acceleration Methods to improve VLA's efficiency usually focus on general quantization or pruning methods. For example, [18, 39, 36] explores quantization techniques applied to VLAs. [46, 59, 61, 60] explored pruning unimportant tokens and layers according to some heuristics. Although reducing the computational complexity of VLAs, these methods just migrate acceleration methods from existing general machine learning models, thus totally ignoring the unique characteristics of VLAs. Different from these methods, recent works [57, 23, 50] explore temporal redundancy in decision making: consecutive frames share temporal correlation, enabling avoiding recomputation of some components by, for example, leveraging the KV-cache of previous steps. However, these methods rely on non-learnable heuristics to recache or simply reuse the previous action, which may suffer from suboptimal performance. Notably, some of these methods [57, 23] implicitly assume that visual similarity in pixel space implies invariance in the latent representations produced by the vision encoder and LLM—an assumption that does not generally hold in transformer-based architectures, as illustrated in Figure 2 and elaborated in Appendix G.

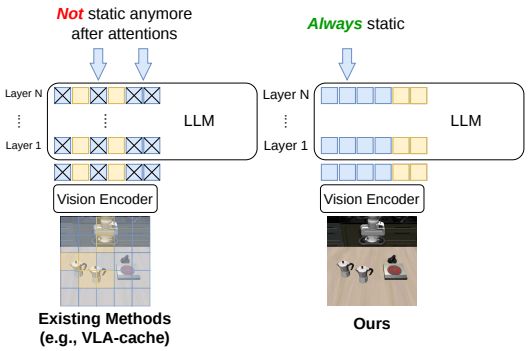


Figure 2: Some of the existing methods [57, 23] that exploit to reuse information in previous frames ignore the problem that the static patches will be affected by attentions even if they are identical in the original pixel space.

Multi-Frame VLAs Existing approaches to incorporating historical frames in VLAs typically provide the model with only indirect or compressed access to temporal information. For example, Jang et al. [15] uses a non-learnable pooling operation to condense historical frames into a fixed number of tokens. Liu et al. [23], Torne et al. [51] incorporate historical frames into the current input via patch-wise mixing. TraceVLA [62] overlays visual traces of keypoints onto the current observation to indicate object trajectories. In addition, MemoryVLA [45] exposes the LLM backbone to only a single frame at each timestep and delegates multi-frame reasoning to a lightweight decoding module, which limits the model's capacity for joint temporal decision-making. In contrast to these methods,

our approach allows the LLM backbone to directly reason over multiple frames without information loss, by explicitly disentangling static and dynamic components and avoiding redundant involvement of temporally persistent visual information.

3 Method

In this section, we first present the problem formulation. We then introduce the architecture of our model, followed by the training objectives that enable static–dynamic disentanglement and the recaching mechanism. Next, we provide a computational complexity analysis and derive the theoretical acceleration and context-length improvements. Finally, we introduce the benchmark for evaluating multi-frame integration.

Problem Formulation Given a timestep t, the corresponding observation of the environment is denoted by $\boldsymbol{X}_t$. A VLA π predicts actions of the following steps based on previous T observations and a language instruction $\mathcal{I}$.

$$\pi : (\boldsymbol{X}_{t-T}, \boldsymbol{X}_{t-T+1}, \cdots, \boldsymbol{X}_t, \mathcal{I}) \mapsto \boldsymbol{a}_t, \boldsymbol{a}_{t+1}, \cdots, \tag{1}$$

where $\boldsymbol{a}_t$ is the action at the t step. Most existing VLAs usually consider $T = 0$ due to the limited context length of large language models.

3.1 Model Architecture

Static-dynamic disentanglement Standard VLA vision backbones typically encode the image $\boldsymbol{X}_t$ into N image tokens $\boldsymbol{Z}_t = \boldsymbol{z}_{t,1}, \boldsymbol{z}_{t,2}, \cdots, \boldsymbol{z}_{t,N}$ as inputs of the VLA's LLM backbone π_{LLM}. Such a design implicitly assumes that all visual tokens must be recomputed and reprocessed at every timestep. However, real-world environments exhibit strong temporal redundancy: many visual attributes remain unchanged across time, while only a subset varies dynamically. Moreover, static information exists at different temporal scales. For example, global scene layout or background structure may persist for long horizons, whereas object-level appearance may change more frequently due to occlusion or interaction. To reflect this structure, we disentangle visual tokens into multi-level static tokens and dynamic tokens. Our design is to split first and learn afterwards.

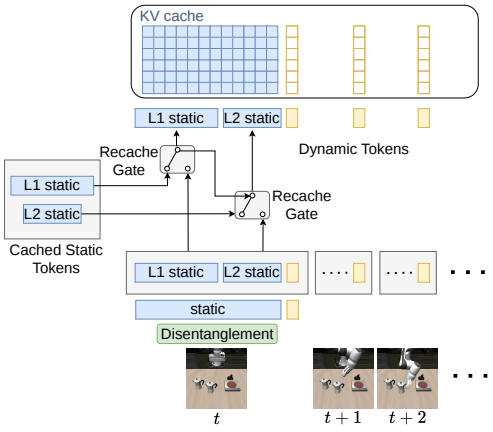


Figure 3: Model architecture overview. We illustrate the design using two levels of static cache. At each level, a recache gate determines whether the cached static tokens should be reused or refreshed. If the L1 cache is refreshed, the L2 cache is also forcibly refreshed.

$$\boldsymbol{Z}_t^{s_1}, \boldsymbol{Z}_t^{s_2}, \cdots, \boldsymbol{Z}_t^{d} = \underbrace{\boldsymbol{z}_{t,1}, \cdots, \boldsymbol{z}_{t,N_{s_1}}}_{\text{static tokens (level 1)}}, \underbrace{\boldsymbol{z}_{t,N_{s_1}+1}, \cdots, \boldsymbol{z}_{t,N_{s_1}+N_{s_2}}}_{\text{static tokens (level 2)}}, \cdots, \underbrace{\boldsymbol{z}_{t,N-N_d+1}, \cdots, \boldsymbol{z}_{t,N}}_{\text{dynamic tokens}} \tag{2}$$

where $\boldsymbol{Z}_t^{s_l}$ denotes static tokens at level l, and $\boldsymbol{Z}_t^{d}$ are dynamic tokens. The ratio between static and dynamic tokens is set as a predefined hyperparameter. Under this design, dynamic tokens capture timestep-specific information and are recomputed at every timestep, whereas static tokens at each level are selectively reused through a learned caching mechanism, described later.

The partition is fixed in advance by token index and stays the same at every timestep. The static slots are then trained to be temporally persistent in the representation space. Position IDs are therefore fixed throughout a rollout and need no special handling.

Temporal dependency modeling Most VLAs [18, 20, 37] are only able to process the current observation ($T = 0$),

$$\boldsymbol{a}_t, \boldsymbol{a}_{t+1}, \cdots = \pi_{\text{LLM}}(\boldsymbol{Z}_t), \tag{3}$$

where π_{LLM} is the LLM backbone of the VLA. With successful static-dynamic disentanglement, we only need to keep *one copy* of the static tokens within a period and construct the input from multiple observations as

$$\boldsymbol{a}_t, \boldsymbol{a}_{t+1}, \cdots = \pi_{\text{LLM}}(\boldsymbol{Z}^{s_1}, \boldsymbol{Z}^{s_2}, \cdots, \boldsymbol{Z}^d_{t-T}, \cdots, \boldsymbol{Z}^d_t), \tag{4}$$

This formulation enables the model to leverage longer temporal context without duplicating static information, effectively alleviating context window bottlenecks while preserving relevant visual cues.

Learning when to recache A key challenge is deciding when cached static tokens should be refreshed. Naively recomputing static tokens at every timestep negates any computational benefit, while overly aggressive reuse of static tokens risks stale representations. To address this, we introduce a learned recache gate at each static level l,

$$g_l(\boldsymbol{Z}_{t-\Delta}, \boldsymbol{Z}_t) \in [0, 1], \quad \Delta = 1, 2, \cdots, \tag{5}$$

which predicts the probability that static tokens should be recomputed given the current observation and a cached reference from Δ timesteps earlier. During training, we use the Gumbel-softmax trick [13, 30] to allow end-to-end differentiable binary decisions. At inference time, static tokens are refreshed if the probability is greater than a threshold, i.e., $g_l > \delta_l$; otherwise, the previous cache is reused. It is also worth noting that if the higher-level cache (e.g., L1) should be refreshed, the lower-level cache (e.g., L2) should also be refreshed during both training and inference. Additional details of the architecture of the recache gate could be found in Appendix B.

Computational and Acceleration Analysis Let N be the total number of tokens per observation, r the fraction of cached static tokens, and T the number of observations in the context. Our static-dynamic disentanglement reduces the effective context length from NT to $NT - rN(T-1)$, and reduces the FLOPs of the LLM backbone by a factor of $1 - r$. The recache gate adds only negligible overhead ($\approx 1.27\%$). A full derivation and wall-clock latency breakdown are provided in Appendix F.

3.2 Training Objective

Besides the standard task loss of the underlying VLA base model $\mathcal{L}_{\text{task}}$, we introduce two additional training objectives. The first promotes temporally persistent static tokens, and the second trains the cache gate that adaptively determines when cached representations should be refreshed.

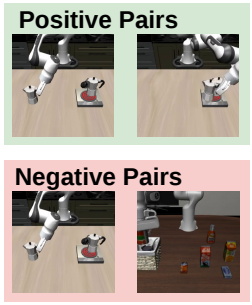


Figure 4: Contrastive loss used to train static tokens to be temporally persistent. Observations from the same trajectory form positive pairs, while observations from different trajectories form negative pairs.

Learning static tokens As illustrated in Figure 4, to ensure that static tokens can be safely and effectively reused across time, they must remain stable over a temporal window while still encoding task-relevant information. To encourage this property, we apply a contrastive term to the static tokens. Specifically, observations from different timesteps within the same trajectory are treated as positive pairs, while observations from different trajectories are treated as negative pairs. We employ InfoNCE loss [35] $\mathcal{L}^l_{\text{InfoNCE}}$ to perform contrastive learning for static tokens at level l.

Training the recache gate If the recache gate is trained only through task supervision, it tends to recompute static tokens at every timestep, which eliminates any computational benefit. To discourage this trivial solution, we add a regularization term that biases the gate toward reuse when observations are close in time.

$$\mathcal{L}_{\text{gate}} = -p_\Delta \log g - (1 - p_\Delta) \log(1 - g), \tag{6}$$

where $p_\Delta = 1 - e^{-\lambda\Delta}$ is a predefined prior which allows the use of current observation only when Δ is large. The recache gate learns to adaptively refresh static tokens only when necessary, balancing computational efficiency and model performance. Note that since λ and the recache threshold δ_l jointly determine the recaching frequency, we fix λ and expose only δ_l as the operating-point knob.



Figure 5: An example of the procedure of the proposed benchmark.

The full training objective is

$$\mathcal{L} = \mathcal{L}_{\text{Task}} + \sum_{l} \alpha_l \mathcal{L}^{l}_{\text{InfoNCE}} + \beta \mathcal{L}_{\text{gate}}, \tag{7}$$

where α_l and β weight the auxiliary regularization terms. A sensitivity analysis for these coefficients is provided in Appendix I.1.

3.3 LIBERO-Memory Benchmark

Existing VLA simulation benchmarks [22, 21] are designed for memoryless tasks, which do not require multi-frame integration, and the current observation alone is theoretically sufficient to predict the next action. For example, in a task with the goal "put X in the basket", [22] the agent can act optimally without retaining information from previous frames. As a result, it is unclear whether the reported performance of existing models for modeling temporal dependency [15, 45] truly stems from improved temporal modeling or from other artifacts. This highlights the absence of a strong and fair benchmark for temporal dependency modeling and multi-frame integration.

To address this gap, we introduce a new simulation benchmark designed to test multi-frame integration explicitly. Our benchmark includes tasks that require a robot to retain and utilize information from past observations, thereby directly evaluating a model's temporal dependency modeling ability.

Following the setup of LIBERO, the robot operates in a tabletop environment containing three objects: two visually distinct cans and a stove. As illustrated in Figure 5, each episode consists of three tasks which reflects the structure of episodic memory [52, 6].

Tasks Each trial consists of three subtasks, each intentionally designed to require the model to retain and reason over information from previous observations at different aspects: 1) Grasp one of the cans, as specified by the instruction, and place it on the stove to heat. 2) After a period of time (given in the instructions), remove the can from the stove and return it to its original location. 3) Once the position of the first can has been restored, grasp the remaining can and place it on the stove.

This task design requires the model to retain *episodic memory*, integrating information about what happened, where it occurred, and when it took place [52, 6]. First, the robot must remember the initial spatial layout in order to return the first can to its original position (where). Second, it must track the elapsed time since the can was placed on the stove to determine when to remove it (when). Third, it must remember which can has already been heated in order to correctly select the remaining one for the final step (what).

These requirements prevent the task from being solved using only the current observation or a short observation history. We use Robosuite [69] as the simulation framework to produce the dataset. The layout and the oracle demonstrations are programmatically generated. More details can be found in Appendix C.

4 Experiments

In this section, we evaluate the model's inference efficiency as well as its multi-frame integration ability. We further present ablation studies of the key components of our method and analyze the model's behavior through attention-map visualizations.

Table 1: Results on the SimplerEnv benchmark. Task suites follow the setting of Qu et al. [37]. All baselines and our method are applied to the same VLA base model (i.e., CogACT [20]).

Method	Visual Matching			Variant Aggregation			Acceleration		
	Pick Can	Move Near	Drawer	Pick Can	Move Near	Drawer	FLOPs	Latency (ms)	Speedup
CogACT [20]	91.3	<u>85.0</u>	71.8	89.6	<u>80.8</u>	28.3	100%	1360	1.00×
+ FlashVLA [50]	80.0	57.1	<u>73.6</u>	82.5	60.8	28.6	83.4%	1024	1.33×
+ TTF [23]	89.3	71.3	58.4	88.0	62.7	<u>34.0</u>	86.5%	1051	1.29×
+ VLA-Cache [57]	<u>92.0</u>	83.3	70.5	<u>91.7</u>	79.3	32.5	80.1%	985	1.38×
+ DySta (Ours)	**92.7**	**88.8**	**75.0**	**92.4**	**81.0**	**38.9**	**43.4%**	**601**	**2.26×**

Table 2: Results on the LIBERO benchmark. Policy inputs: third-person image, language instruction. All baselines and our method are applied to the same VLA base model (i.e., OpenVLA-OFT [19]).

Models	Spatial	Object	Goal	Long	Average	FLOPs	Latency (ms)	Speedup
OpenVLA-OFT [19]	<u>96.2</u>	**98.3**	**96.2**	<u>90.7</u>	<u>95.4</u>	100.0%	741	1.00×
+ FlashVLA [50]	71.8	90.0	79.2	78.8	80.0	88.7%	657	1.13×
+ TTF [23]	95.8	94.2	93.6	86.0	92.4	75.0%	575	1.29×
+ VLA-Cache [57]	95.6	92.8	94.0	89.6	93.0	85.5%	652	1.14×
+ DySta (Ours)	**97.8**	<u>97.6</u>	**96.2**	**92.4**	**96.0**	**63.4%**	**437**	**1.70×**

4.1 Acceleration

In this study, we validate our method's performance under acceleration by enabling KV caching of static components.

Baselines For baselines, we compare to those that also adopt temporal information reuse. *FlashVLA* [50] simply reuses the previous action based on a heuristic criterion. *TTF-VLA* [23] and *VLA-cache* [57] identify reusable visual patches using hand-designed heuristics and cache the corresponding embeddings from previous steps; however, these cached patches are not guaranteed to remain static in the hidden representation space. All methods are based on the same VLA base model (i.e., CogACT [20]), with all other necessary variables controlled consistently. The acceleration baselines in Table 1 use their training-free protocols. We additionally retrain TTF and VLA-Cache from the released CogACT checkpoint with the same training configuration as DySta (Table 12).

Simulation benchmark We use SimplerEnv [21] and LIBERO [22] task suite to evaluate the performance and acceleration. Kim et al. [19] offers two settings for LIBERO, one using only a third-person image and a language instruction as inputs, and another that additionally includes an extra camera view and proprioceptive state. We adopt the former setting, as it offers broader applicability and aligns more closely with standard VLA deployment scenarios. For the LIBERO benchmark, we finetune the model to the corresponding downstream datasets. For the SimplerEnv benchmark, we use Open X-Embodiment (OXE) [54] as the training dataset. All settings are consistent with the corresponding original base model. Dataset statistics could be found in Appendix A.

Real-World benchmark We evaluate DySta with Cobot Magic robots on 7 tasks across 3 categories, each with 20 trials. This evaluation scale is consistent with prior work [45, 20]. Models are finetuned on each task. More details could be found in Appendix A.2.

Evaluation The performance is measured by success rate, and acceleration is measured by the FLOPs reduction rate and the inference latency.

Implementation details We use two levels of static cache. L1 static cache consists of 133 (52%) tokens and L2 static cache consists of 107 (42%) tokens, resulting in 16 dynamic tokens. The sensitivity analysis of the static ratio is in Appendix I.2. FLOPs are measured by open-source tools [2]. Additional details, including the recaching threshold δ_l, the resulting average recaching intervals and the latency distribution when rollout, can be found in Appendix D and Appendix H.

[2] `https://github.com/MrYxJ/calculate-flops.pytorch`

Table 3: Success rate (%) on real-world experiments with the Cobot Magic robot. All baselines and our method are applied to the same VLA base model (i.e., OpenVLA-OFT [19]).

Method	Pick-and-Place				Pouring		Container	Acceleration Metrics		
	Corn →Pot	Corn →Plate	Bread →Pan	Carrot Transfer	Pour Coke	Pour Bottle	Open Drawer	FLOPs	Latency (ms)	Speedup
OpenVLA-OFT	55	<u>70</u>	**55**	<u>65</u>	<u>60</u>	35	**35**	100.0%	1089	1.00×
+ FlashVLA	35	55	30	50	55	30	15	<u>62.5%</u>	<u>791</u>	<u>1.30×</u>
+ TTF	35	<u>70</u>	30	55	55	<u>40</u>	15	98.6%	1077	1.01×
+ VLA-Cache	<u>60</u>	65	35	50	45	<u>40</u>	30	74.0%	836	1.27×
+ DySta (Ours)	**65**	**80**	<u>50</u>	**70**	**60**	**50**	**35**	**30.5%**	**493**	**2.21×**

Results Table 1 and Table 2 summarize the performance and the acceleration results. Table 3 summarizes the results on the real-world benchmark. Across tables, we show that our method achieves improved or comparable performance across benchmarks with substantial speedup. Notably, on the SimplerEnv benchmark, our method improves the base model by 4.9%, and outperforms the best baseline by 3.9% and achieves $2.26\times$ acceleration. On the real-world benchmark, our model improves the base model by 10.6% and achieves $2.21\times$ acceleration.

Training-matched comparisons retain a 3.4% relative mean-success advantage over the strongest retrained baseline. Appendix K also reports results on SpatialVLA, wrist-camera observations, and LIBERO-Plus. These standard-benchmark results test acceleration; the following evaluation tests multi-frame integration.

4.2 Multi-Frame Integration for Temporal-dependency Modeling

Baselines We consider the following baselines: *OpenVLA-OFT*† incorporates full image tokens from historical observations. Since this baseline incurs a prohibitively large context length compared with standard models, we allocate the maximum available resources and tune it for the best achievable performance. *TTF-VLA* [23] mixes the current observation with past observations in a patch-wise manner. *TraceVLA* [62] visualizes the trajectories of active points by overlaying them onto the image using distinct colors, referred to as visual traces. *MemoryVLA* [45] feeds the LLM backbone with images from each timestep independently and extracts a representation for each; only the multi-timestep representations are provided to the decoder. *ContextVLA* [15] pools historical observations into a fixed number of tokens and prepends them to the image tokens of the current observation. All methods are applied to the same VLA base model (i.e., OpenVLA-OFT [19]), with all other necessary variables controlled consistently.

Simulation benchmark and evaluation We adopt the proposed LIBERO-memory benchmark to evaluate multi-frame integration by memory-dependent tasks, whose designs are inspired by the notion of episodic memory. The benchmark comprises three objectives. (1) *(where) Position Reset.* After heating the first can, the robot is required to remember its original position and return it to the position. Performance is measured by success rate, where an episode is considered successful if the positional error is below a threshold. (2) *(when) Doneness.* The model must track the elapsed heating time and remove the can at the appropriate moment. Given a target heating duration specified in the instruction, we measure the absolute difference (in seconds) between the actual and desired heating times, capturing overcooking or undercooking. (3)*(what) On-Stove.* After heating and resetting the first can, the robot is required to remember which can has been cooked and place the second can on the stove for heating. We report the success rate of placing the second can on the stove.

Real-world benchmark We evaluate DySta with Cobot Magic robots on 3 tasks, each with 20 trials. Models are fine-tuned on each task. Performance is measured by success rate. More details could be found in Appendix A.2.

Table 4: Results on the temporal-dependent tasks for multi-frame integration. All models use OpenVLA-OFT [19] as the base model. OpenVLA-OFT† denotes a naive baseline that packs as many frames as possible into the context.

Models	On Stove ↑	Position Reset ↑	Doneness (seconds) ↓
OpenVLA-OFT†	39.0%	78.0%	0.44
TTF-VLA [23]	7.8%	1.5%	1.51
TraceVLA [62]	2.0%	3.0%	1.41
MemoryVLA [45]	23.0%	2.0%	1.49
ContextVLA [15]	50.8%	22.3%	0.37
DySta (Ours)	**69.8%**	**83.0%**	**0.26**

Implementation details We use one static level. The ratio of static tokens is set to 0.9, corresponding to 230 static tokens and 26 dynamic tokens. The number of observations is set to 20, with a sampling interval of 20 frames (1 frame per second), resulting in 750 context tokens. More details could be found in Appendix D. For the temporal comparisons, all methods receive 20 observations sampled at 1 Hz except OpenVLA-OFT†, which uses the largest full-token window we could train: six observations (1,536 visual tokens). Table 13 reports per-method histories and latency.

Results As shown in Table 4, our method significantly and consistently outperforms all baselines on all objectives, with an average improvement of 24.5%. On the real-world experiments in Appendix E, we achieve 23.3% absolute in absolute success rate. Single-image methods such as TraceVLA almost always fail, since the task cannot be solved from the current observation alone, and visual traces provide only marginal additional benefit. ContextVLA underperforms due to non-learnable pooling observations, which may discard critical temporal and spatial details. MemoryVLA presents the LLM backbone with only a single frame at each timestep, leaving multi-frame reasoning to a lightweight decoder; as a result, the LLM itself does not reason over multiple frames, which limits the model's capacity. While these baselines have their own ways to reduce context length, they do so at the cost of information or model capacity. In contrast, our approach preserves all relevant visual information by reusing temporally persistent static tokens, enabling effective multi-frame integration within a limited context window.

4.3 Ablation Studies

To validate the effectiveness of the proposed components, we conduct ablation studies on each of them, with results summarized in Table 5. From the second row, removing the contrastive learning objective leads to a noticeable performance degradation, as there is no longer an explicit mechanism to enforce temporal consistency in the static tokens. Performance also declines when the L2 static cache is removed and only a single-level static cache is used, as shown in the third row, highlighting the importance of the proposed multi-level caching design. Finally, replacing the learnable recache gate results in a further drop in performance, demonstrating the necessity of adaptive cache refreshing. Appendix K reports additional ablations on the static-token budget and L1/L2 split.

4.4 Visualizations

We visualize the attention between the image and static/dynamic tokens across timesteps in Figure 6.

Table 5: Ablation studies. *w/o Contrast* removes the contrastive learning objective during training. *w/o L2 cache* ablates the multi-level design by retaining only a single static cache level. *Fixed step* replaces the learnable recache gate by forcing the refresh of the cache at fixed intervals. The refresh interval is set to match the average recaching interval of DySta (first row).

	visual matching		variant aggregation	
	PickCan	MoveNear	PickCan	MoveNear
DySta	**92.7**	**88.8**	**92.4**	81.0
w/o Contrast	91.3	84.6	91.6	**82.7**
w/o L2 cache	92.0	85.0	91.5	81.5
fixed step	87.7	81.7	85.3	74.8

Dynamic tokens consistently attend to movable objects, most notably the gripper (including its shadow) and the apple on the tabletop. This behavior aligns with their intended role of capturing temporally varying, action-relevant elements in the scene.

L1 static tokens, which are designed to represent the most persistent visual information, primarily focus on background and

ambient regions. Their attention heatmaps are often strongest in areas without salient foreground objects. We interpret this behavior as L1 static tokens of shallow layers functioning as sink tokens, capturing coarse, global scene context rather than object-specific details. Importantly, their attention patterns remain highly consistent across timesteps, reflecting strong temporal invariance.

L2 static tokens exhibit intermediate behavior between L1 static and dynamic tokens. They attend more strongly to semi-static objects, such as the drawer, which typically remains stationary most of the time. At $t = 0$ both drawer handles are highlighted, as either could potentially be moved. After the drawer is opened ($t > 100$), the attention maps stabilize and remain focused on the drawer structure, indicating that these tokens capture object-level static information with moderate temporal persistence. The robotic arm is also highlighted at certain frames, which we attribute to its importance in the task and the need to preserve its appearance across timesteps.

We further illustrate how static-dynamic assignments evolve along a trajectory in Appendix J.

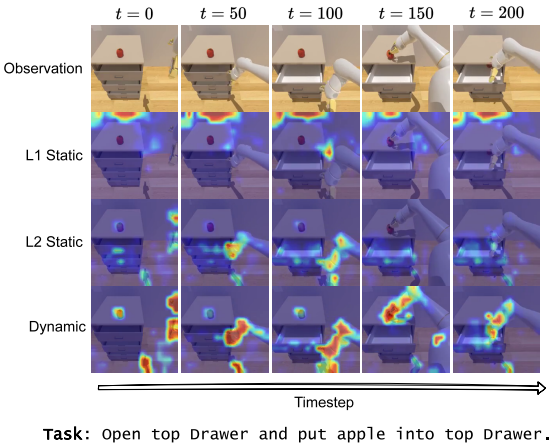


Figure 6: Attention map visualization across time. For each token, we compute its last-layer attention to image patches and upsample the result to the full image resolution to produce a heatmap. Heatmaps are averaged over tokens of the same type and displayed by row (e.g., the Dynamic row shows the average attention heatmap of all dynamic tokens).

5 Conclusion

We introduced DySta, a method that improves efficiency and multi-frame integration in VLA models via static–dynamic disentanglement. By reusing temporally persistent static tokens and selectively refreshing them through a learnable recache gate, DySta enables compact multi-frame contexts and efficient inference. Experiments show that our approach substantially improves performance on temporally dependent tasks while providing meaningful speedups over strong baselines. These results suggest that explicitly modeling temporal persistence is a promising direction for scalable and practical VLA systems. **Limitations**. We realize our approach based on pretrained VLAs. Such a strategy may not fully unleash the performance of the model. Future work could focus on pretraining VLAs with our architecture from scratch.

References

[1] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Hydra: Hybrid robot actions for imitation learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[2] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, et al. Paligemma: A versatile 3b vlm for transfer. *arXiv preprint arXiv:2407.07726*, 2024.

[3] 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, 2024. URL `https://arxiv.org/abs/2410.24164`.

[4] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[5] Lawrence Yunliang Chen, Simeon Adebola, and Ken Goldberg. Berkeley UR5 demonstration dataset. `https://sites.google.com/view/berkeley-ur5/home`.

[6] Nicola S Clayton and Anthony Dickinson. Episodic-like memory during cache recovery by scrub jays. *Nature*, 395(6699):272–274, 1998.

[7] Zichen Jeff Cui, Yibin Wang, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. From play to policy: Conditional behavior generation from uncurated robot data. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2023.

[8] Shivin Dass, Jullian Yapeter, Jesse Zhang, Jiahui Zhang, Karl Pertsch, Stefanos Nikolaidis, and Joseph J. Lim. Clvr jaco play dataset, 2023. URL `https://github.com/clvrai/clvr_jaco_play_dataset`.

[9] Frederik Ebert, Yanlai Yang, Karl Schmeckpeper, Bernadette Bucher, Georgios Georgakis, Kostas Daniilidis, Chelsea Finn, and Sergey Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets. In *Proceedings of Robotics: Science and Systems (RSS)*, 2022.

[10] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jinlan Fu, Jingjing Gong, and Xipeng Qiu. LIBERO-Plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025. URL `https://arxiv.org/abs/2510.13626`.

[11] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. URL `https://arxiv.org/abs/2106.09685`.

[12] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael 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, 2025. URL `https://arxiv.org/abs/2504.16054`.

[13] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[14] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2022.

[15] Huiwon Jang, Sihyun Yu, Heeseung Kwon, Hojin Jeon, Younggyo Seo, and Jinwoo Shin. Contextvla: Vision-language-action model with amortized multi-frame context. *arXiv preprint arXiv:2510.04246*, 2025.

[16] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2018.

[17] Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *Forty-first International Conference on Machine Learning*, 2024.

[18] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[19] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. 2025. URL `https://arxiv.org/abs/2502.19645`.

[20] Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

[21] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[22] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

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

[24] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[25] Huihan Liu, Soroush Nasiriany, Lance Zhang, Zhiyao Bao, and Yuke Zhu. Robot learning on the job: Human-in-the-loop autonomy and learning during deployment. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[26] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL `https://arxiv.org/abs/1711.05101`.

[27] Guanxing Lu, Wenkai Guo, Chubin Zhang, Yuheng Zhou, Haonan Jiang, Zifeng Gao, Yansong Tang, and Ziwei Wang. Vla-rl: Towards masterful and general robotic manipulation with scalable reinforcement learning. *arXiv preprint arXiv:2505.18719*, 2025.

[28] Jianlan Luo, Charles Xu, Xinyang Geng, Gilbert Feng, Kuan Fang, Liam Tan, Stefan Schaal, and Sergey Levine. Multi-stage cable routing through hierarchical imitation learning. *IEEE Transactions on Robotics*, 40:1476–1491, 2024.

[29] Jianlan Luo, Charles Xu, Fangchen Liu, Liam Tan, Zipeng Lin, Jeffrey Wu, Pieter Abbeel, and Sergey Levine. Fmb: a functional manipulation benchmark for generalizable robotic learning. *The International Journal of Robotics Research*, 2024.

[30] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[31] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2018.

[32] Oier Mees, Jessica Borja-Diaz, and Wolfram Burgard. Grounding language with visual affordances over unstructured data. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[33] Russell Mendonca, Shikhar Bahl, and Deepak Pathak. Structured world models from human videos. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[34] Soroush Nasiriany, Tian Gao, Ajay Mandlekar, and Yuke Zhu. Learning and retrieval from prior data for skill-based imitation learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[35] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[36] Seongmin Park, Hyungmin Kim, Wonseok Jeon, Juyoung Yang, Byeongwook Jeon, Yoonseon Oh, and Jungwook Choi. Quantization-aware imitation-learning for resource-efficient robotic control, 2024. URL `https://arxiv.org/abs/2412.01034`.

[37] Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025.

[38] Gabriel Quere, Annette Hagengruber, Maged Iskandar, Samuel Bustamante, Daniel Leidner, Freek Stulp, and Jörn Vogel. Shared control templates for assistive robotics. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[39] Siddharth Reddy, Anca D. Dragan, and Sergey Levine. Sqil: Imitation learning via reinforcement learning with sparse rewards, 2019. URL `https://arxiv.org/abs/1905.11108`.

[40] Erick Rosete-Beas, Oier Mees, Gabriel Kalweit, Joschka Boedecker, and Wolfram Burgard. Latent plans for task-agnostic offline reinforcement learning. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2022.

[41] Saumya Saxena, Mohit Sharma, and Oliver Kroemer. Multi-resolution sensing for real-time control with vision-language models. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[42] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[43] Rutav Shah, Roberto Martín-Martín, and Yuke Zhu. Mutex: Learning unified policies from multimodal task specifications. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[44] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[45] Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation, 2025. URL `https://arxiv.org/abs/2508.19236`.

[46] 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`.

[47] SimpleVLA-RL Team. Simplevla-rl: Online rl with simple reward enables training vla models with only one trajectory. `https://github.com/PRIME-RL/SimpleVLA-RL`, 2025. GitHub repository.

[48] Andreas Steiner, André Susano Pinto, Michael Tschannen, Daniel Keysers, Xiao Wang, Yonatan Bitton, Alexey Gritsenko, Matthias Minderer, Anthony Sherbondy, Shangbang Long, et al. Paligemma 2: A family of versatile vlms for transfer. *arXiv preprint arXiv:2412.03555*, 2024.

[49] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[50] Xudong Tan, Yaoxin Yang, Peng Ye, Jialin Zheng, Bizhe Bai, Xinyi Wang, Jia Hao, and Tao Chen. Think twice, act once: Token-aware compression and action reuse for efficient inference in vision-language-action models. *arXiv preprint arXiv:2505.21200*, 2025.

[51] Marcel Torne, Karl Pertsch, Homer Walke, Kyle Vedder, Suraj Nair, Brian Ichter, Allen Z. Ren, Haohuan Wang, Jiaming Tang, Kyle Stachowicz, Karan Dhabalia, Michael Equi, Quan Vuong, Jost Tobias Springenberg, Sergey Levine, Chelsea Finn, and Danny Driess. Mem: Multi-scale embodied memory for vision language action models, 2026. URL `https://www.pi.website/download/Mem.pdf`.

[52] Endel Tulving et al. Episodic and semantic memory. *Organization of memory*, 1(381-403):1, 1972.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[54] Quan Vuong, Sergey Levine, Homer Rich Walke, Karl Pertsch, Anikait Singh, Ria Doshi, Charles Xu, Jianlan Luo, Liam Tan, Dhruv Shah, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition@ CoRL2023*, 2023.

[55] Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[56] Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Song XiXuan, et al. Cogvlm: Visual expert for pretrained language models. *Advances in Neural Information Processing Systems*, 37:121475–121499, 2024.

[57] Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Towards efficient vision-language-action model via adaptive token caching in robotic manipulation. *arXiv preprint arXiv:2502.02175*, 2025.

[58] Ge Yan, Kris Wu, and Xiaolong Wang. ucsd kitchens dataset. `https://github.com/geyan21/rlds_dataset_builder/tree/main/ucsd_kitchens`, 2023.

[59] 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. *arXiv preprint arXiv:2506.10100*, 2025.

[60] Yang Yue, Yulin Wang, Bingyi Kang, Yizeng Han, Shenzhi Wang, Shiji Song, Jiashi Feng, and Gao Huang. Deer-vla: Dynamic inference of multimodal large language models for efficient robot execution. *Advances in Neural Information Processing Systems*, 37:56619–56643, 2024.

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

[62] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

[63] Yifan Zhong, Fengshuo Bai, Shaofei Cai, Xuchuan Huang, Zhang Chen, Xiaowei Zhang, Yuanfei Wang, Shaoyang Guo, Tianrui Guan, Ka Nam Lui, Zhiquan Qi, Yitao Liang, Yuanpei Chen, and Yaodong Yang. A survey on vision-language-action models: An action tokenization perspective, 2025. URL https://arxiv.org/abs/2507.01925.

[64] Gaoyue Zhou, Victoria Dean, Mohan Kumar Srirama, Aravind Rajeswaran, Jyothish Pari, Kyle Hatch, Aryan Jain, Tianhe Yu, Pieter Abbeel, Lerrel Pinto, et al. Train offline, test online: A real robot learning benchmark. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[65] Xinghao Zhu, Ran Tian, Chenfeng Xu, Mingxiao Huo, Wei Zhan, Masayoshi Tomizuka, and Mingyu Ding. Fanuc manipulation: A dataset for learning-based manipulation with fanuc mate 200id robot. https://sites.google.com/berkeley.edu/fanuc-manipulation, 2023.

[66] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models, 2024. URL https://arxiv.org/abs/2308.07633.

[67] Yifeng Zhu, Peter Stone, and Yuke Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2): 4126–4133, 2022.

[68] Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

[69] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Kevin Lin, Abhiram Maddukuri, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning, 2025. URL https://arxiv.org/abs/2009.12293.

A Dataset and Benchmark Details

A.1 Simulation

Training Table 6 summarizes the Open-X-Embodiment dataset we use for the base model CogACT [20]. Table 7 summarizes the LIBERO dataset we use for the base model OpenVLA-OFT [19].

Table 6: Open-X-Embodiment dataset composition we use for CogACT training.

Dataset	Ratio
Fractal [4]	27.1%
Kuka [16]	14.7%
Bridge [55, 9]	15.3%
Taco Play [40, 32]	3.4%
Jaco Play [8]	0.6%
Berkeley Cable Routing [28]	0.3%
Roboturk [31]	2.7%
Viola [68]	1.1%
Berkeley Autolab UR5 [5]	1.4%
Toto [64]	2.3%
Stanford Hydra Dataset [1]	5.1%
Austin Buds Dataset [67]	0.2%
NYU Franka Play Dataset [7]	1.0%
UCSD Kitchen Dataset [58]	<0.1%
Austin Sailor Dataset [34]	2.5%
Austin Sirius Dataset [25]	2.0%
DLR EDAN Shared Control [38]	<0.1%
IAMLab CMU Pickup Insert [41]	1.0%
UTAustin Mutex [43]	2.6%
Berkeley Fanuc Manipulation [65]	0.9%
CMU Stretch [33]	0.2%
BC-Z [14]	8.6%
FMB Dataset [29]	2.4%

Table 7: LIBERO dataset

Dataset	Libero-Spatial	Libero-Object	Libero-Goal	Libero-Long
#(trajectories)	432	454	428	379

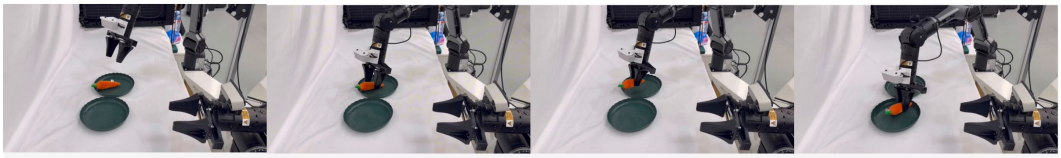
SimplerEnv SimplerEnv [21] is a simulation-based benchmark designed for tabletop robotic manipulation. It is explicitly constructed to minimize the sim-to-real gap, demonstrating strong alignment between simulation performance and real-world execution across multiple robot platforms. The benchmark supports two complementary evaluation configurations: Visual Matching, which emphasizes high visual fidelity to real-world environments, and Variant Aggregations, which systematically introduce visual perturbations such as changes in background, lighting, distractors, and surface textures to assess robustness. SIMPLER includes tasks instantiated on both the Google Robot and the WidowX robot, covering a diverse set of manipulation primitives such as picking, placing, navigation, and articulated object interaction. In all tasks, agents receive RGB visual observations and natural language instructions. SIMPLER offers two evaluation settings:

- *Visual Matching:* Real-world images are overlaid onto simulated environments, with foreground objects and robots adjusted to closely match real-world appearances.
- *Variant Aggregation:* Multiple environmental variations are generated—such as different backgrounds, lighting conditions, and surface textures.

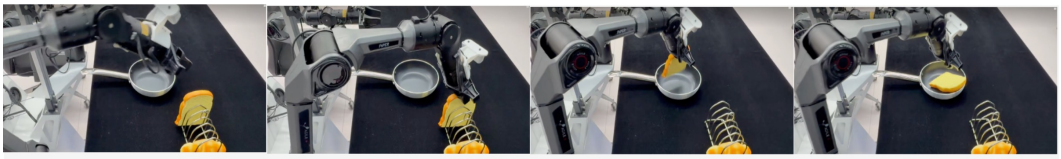
LIBERO LIBERO is built around a simulated Franka Emika Panda arm and provides high-quality human-teleoperated demonstration data paired with language-conditioned tasks to support sample-



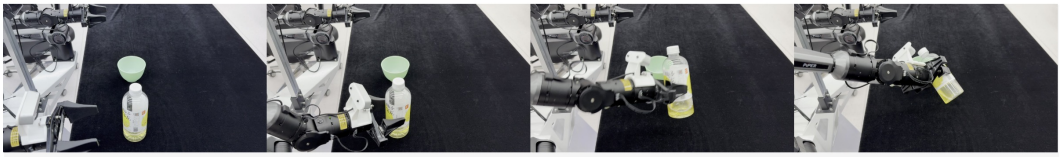
Put the corn on the plate.



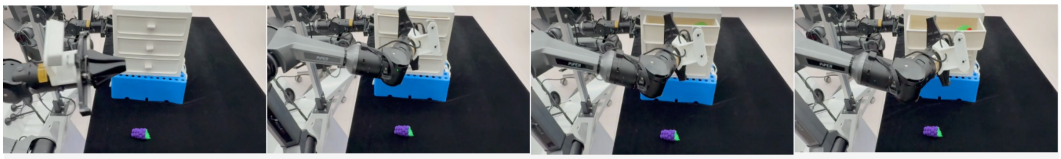
Transfer the carrot to another plate.



Put the bread into the pan



pour the bottle into the bowl.



open the top drawer

Figure 7: In real-world evaluations with the Cobot Magic robot, each task is evaluated over 20 episodes.

efficient learning and generalization. The benchmark comprises four distinct task suites—LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-Long, each providing 10 distinct tasks.

A.2 Real-World

For the acceleration experiments, we include 7 tasks across 3 categories in the acceleration experiment, with 20 episodes used for training and 20 trials used for evaluating each task. The task designs are detailed below:

- **Pick-and-Place.** The robot is instructed to move an object from its original location to a specified destination (e.g., a pan or pot). We report the success rate for each object.
- **Pouring.** The robot is instructed to pour the contents of a bottle or can into a container. We report the success rate for each target container.
- **Drawer Opening.** The robot is instructed to open a specified drawer. We report the success rate of successfully opening the drawer.

For the multi-frame integration experiments, we evaluate three tasks that require memory or temporal reasoning:



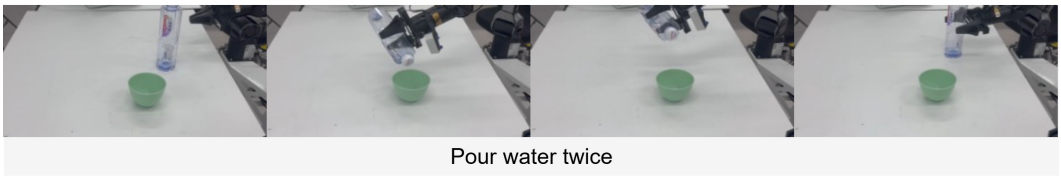




Figure 8: In real-world evaluations with the Cobot Magic robot for multi-frame integration, each task is evaluated over 20 episodes.

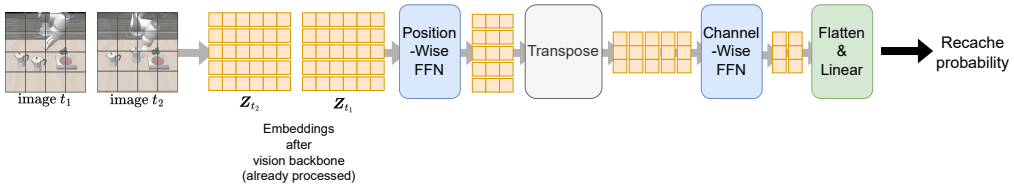


Figure 9: Architecture of the recache gate.

- **Sequential Heat.** The robot is instructed to heat the vegetables one at a time. The model must remember which vegetables have already been heated.
- **Pour Twice.** The robot is instructed to pour from the bottle twice. The model must keep track of whether the first pour has already been completed.
- **Press Buttons.** The robot is instructed to press a sequence of buttons. The model must remember which buttons have already been pressed and determine the next button to press.

For each task, we vary both the object's location and the robotic arm's starting position. The demonstrations of tasks are illustrated in Figure 8 and Figure 10.

B Architecture Details

Recache gate The recache gate is designed to predict the probability of refreshing cached representations based on images (or their latent representations) observed at different timesteps. To minimize additional inference latency in the VLA, the recache gate must remain lightweight. Accordingly, we adopt a simple MLP-based architecture.

As illustrated in Figure 9, the gate takes as input latent embeddings $\boldsymbol{Z}_{t_1}$ and $\boldsymbol{Z}_{t_2}$ produced by the VLA's vision backbone at two different timesteps. We first apply a position-wise feedforward network to each embedding. The embeddings are then transposed to enable a channel-wise feedforward operation. After these transformations and dimensionality reduction, the resulting features are flattened and passed through a final MLP to predict the recache probability. For different static levels, we share all parameters except for the final prediction head.

C LIBERO-Memory Benchmark

Following the LIBERO benchmark [22], the layout initialization is specified by the BDDL scene description file, which is a file format that specifies the initial layout's sampling strategy and other meta information.

```
(define (problem LIBERO_Kitchen_Tabletop_Manipulation)
  (:domain robosuite)
  (:language Heat the tomato sauce on the stove for 1.4 second and then heat
    the alphabet soup)
    (:regions
      (operation_region
          (:target kitchen_table)
          (:ranges (
              (-0.2 0.09999999999999999 0.0 0.2)
            )
          )
          (:yaw_rotation (
              (0.0 0.0)
            )
          )
      )
      (stove_region
          (:target kitchen_table)
          (:ranges (
              (-0.30000000000000004 -0.2 -0.1 -0.09999999999999999)
            )
          )
          (:yaw_rotation (
              (0.0 0.0)
            )
          )
      )
      (cook_region
          (:target flat_stove_1)
      )
    )

  (:fixtures
    kitchen_table - kitchen_table
    flat_stove_1 - flat_stove
  )

  (:objects
    alphabet_soup_1 - alphabet_soup
    tomato_sauce_1 - tomato_sauce
  )

  (:obj_of_interest
    tomato_sauce_1
    alphabet_soup_1
  )

  (:init
    (On tomato_sauce_1 kitchen_table_operation_region)
    (On alphabet_soup_1 kitchen_table_operation_region)
    (On flat_stove_1 kitchen_table_stove_region)
    (Turnon flat_stove_1)
  )

  (:goal
    (And (CloseXY tomato_sauce_1) (On alphabet_soup_1
    flat_stove_1_cook_region))
  )

)
```

Listing 1: An example of the PDDL scene description file used in LIBERO-Memory.

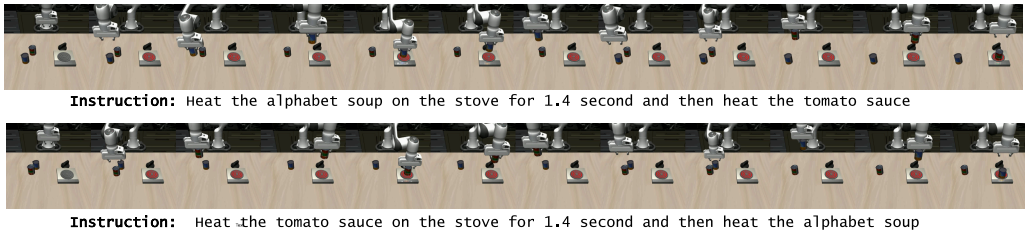


Figure 10: Examples of our proposed LIBERO Memory Benchmark.

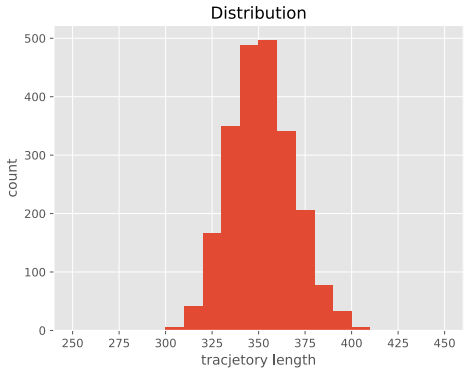


Figure 11: Distribution of the trajectory length of the LIBERO-Memory benchmark

Oracle demonstrations We generate expert trajectories programmatically because the correct behavior in LIBERO-Memory is fully specified and unambiguous. For example, for a sub-task such as "place the tomato sauce can on the stove", an oracle with access to the simulator state can deterministically execute the correct sequence. The oracle knows the target object and its current location, and issues movement commands that drive the end-effector toward that location. After each movement, it compares the updated end-effector position with the desired target and decides the next control direction. Once the end-effector is sufficiently close, the oracle closes the gripper to grasp the object and subsequently moves it to the specified target location.

Figure 10 shows more detailed example trajectories of the benchmark. Figure 11 shows the distribution of the trajectory length of the benchmark.

To better contextualize the absolute performance of our model, we include a human baseline for the benchmark in Table 8. Specifically, we recruited two human participants familiar with the task setup, who performed the tasks using standard keyboard controls. Each human participant completed 50 rollout episodes. The performance varies across subtasks: *On Stove* is largely semantic and can be solved reliably by humans, while *Position Reset* and *Doneness* require precise tracking of spatial locations and elapsed time over long horizons, making them more challenging for humans and highlighting greater potential for machine learning models to excel.

Table 8: Human Performance on LIBERO-Memory.

Models	On Stove ↑	Position Reset ↑	Doneness ↓
Human 1	98.0%	74.0%	1.58
Human 2	96.0%	68.0%	1.44
DySta (Ours)	69.8%	83.0%	0.26

D Implementation Details and Hyperparameter Choices

For the recache gate in Figure 9, the first FFN is a 3-layer MLP that reduce the embedding dimension from 2×4096 to 128. The second FFN is a 3-layer MLP that reduce the number of embeddings from 256 to 128. The head after the flatten operation is a linear layer with input dimension 128×64.

CogACT We use LoRA [11] to train the model, with the rank of 32. We use AdamW [26] with a learning rate of 2×10^{-5}. Training is conducted on $2\times$ H100 for 6000 steps, with a global batch size of 32.

OpenVLA-OFT We use LoRA [11] to train the model, with the rank of 32. We use AdamW [26] with a learning rate of 5×10^{-4}. Training is conducted on $2\times$ H100 for 15000 steps, with a global batch size of 64.

For both base models, the coefficients of the training objectives are set as $\alpha_1 = 0.2, \alpha_2 = 0.1, \beta = 0.1$. For their corresponding benchmarks, the evaluation and the acceleration measurements are all conducted on $1\times$ L40S for each task.

Table 9 summarizes the recaching threshold settings and the resulting average recaching intervals.

Table 9: Recaching threshold settings and the resulting average recaching intervals

	Pick Can	Move Near	Drawer
δ_1	0.8	0.4	0.7
δ_2	0.4	0.3	0.4
avg interval of L1	9.14	2.23	7.47
avg interval of L2	2.15	2.05	4.16

E Real-World Experiments on Multi-Frame Integration

We summarize the results of real-world experiments on multi-frame integration in Table 10. We found that our method outperforms baselines, with an average absolute improvement of 23.3%.

F Computational and Acceleration Analysis

In this section, we provide the full computational analysis of the context length reduction and the acceleration enabled by our approach.

Context length Let N be the number of total tokens (including image tokens and text tokens). Let r be the fraction of tokens that are cached, and $N' = (1-r)N$ be the number of recomputed tokens.

Let T be the number of observations that should be incorporated in the context. The context length with our method reduces from NT to $rN + (1-r)NT = NT - rN(T-1)$.

Static token reuse The complexity of the language model backbone in the VLA model mainly comes from two types of modules: the multi-head attention (MHA) layers and the feedforward neural networks. Let d be the dimension of embedding. The FLOPs of the MHA layer are estimated by $4Nd^2 + 2N^2d$, and the FLOPs of the feedforward neural network are estimated by $2Ndm$, where m is the hidden dimension of the MLP. Therefore, the total FLOPs can be written as $F = 4Nd^2 + 2N^2d + 2Ndm$.

With our method, given that the static tokens are cached, only the dynamic parts (including the dynamic image tokens and the language instructions), the projection layers in the MHA layer will reduce to $4N'd^2$, the attention matrix calculation will reduce to $2N'Nd$, and the FLOPs of the feedforward network will reduce to $2N'dm$. The final FLOPs are $F' = 4N'd^2 + 2NN'd + 2N'dm$.

Table 10: Real-world task accuracy (%) on temporally dependent tasks for multi-frame integration. Each task is evaluated over 20 trials.

Method	Heat Vegetables	Pour Twice	Press Buttons	Average
MemoryVLA	10.0	25.0	45.0	26.7
ContextVLA	25.0	40.0	40.0	35.0
DySta (Ours)	**35.0**	**75.0**	**65.0**	**58.3**

Table 11: Breakdown of the CUDA latency of each component.

Component	CUDA Latency (ms)	ratio
Vision Backbone	246.9	41.1%
Recache Gate	18.7	3.1%
LLM Backbone	335.7	55.8%
Total	601.2	100.0%

Therefore, the theoretical FLOPs reduction of the LLM backbone *under an idealized setting* could be estimated as

$$\frac{F'}{F} = \frac{4N'd^2 + 2NN'd + 2N'dm}{4Nd^2 + 2N^2d + 2Ndm} = (1-r)\frac{4d^2 + 2Nd + 2dm}{4d^2 + 2Nd + 2dm} = 1 - r \tag{8}$$

The overall reduction will be diluted by the involvement of the vision backbone, the decoding module, and the cache refresh step. However, these components are significantly less computationally intensive than the LLM backbone. Moreover, the introduction of the recache gate incurs only negligible overhead ($\approx 1.27\%$) and has minimal impact on the overall computational complexity. The latency breakdown of each component is summarized in Table 11.

G Latent Representation Drift

Methods such as [57] implicitly assume similarity in the pixel space lead to similarity in the latent space. In this section, we analyze the latent representations during VLA rollouts to show that this assumption is fundamentally flawed.

As illustrated in Figure 12, we measure cosine similarity between tokens at each timestep and their initial tokens in both pixel and latent space. Visually static tokens still exhibit substantial latent drift — e.g., in layer 0, a pixel-constant token (orange) drops to 0.5 in latent space after just one timestep, while the blue token with minor pixel changes (steps 15–20) drops from 0.5 to 0.2. These results confirm that pixel-level similarity does not imply latent invariance, thus invalidating existing methods and motivating our learned recache mechanism operating in representation space.

H Latency Distribution

We provide the latency distribution during rollout across tasks in Figure 13. On the *Pick Can* task, 86.0% of steps fully reuse static tokens across all levels, resulting in low latency (250 ms). A smaller fraction (7.4%) triggers an L2 refresh (500 ms), while only 6.6% of steps require an L1 refresh, leading to higher latency (900 ms).

These results show that high-latency events are infrequent, with the vast majority of steps operating in the efficient reuse regime. Since our goal is to optimize overall rollout efficiency (in addition to enabling multi-frame integration), occasional latency spikes have a negligible impact on end-to-end performance.

I Hyperparameter Sensitivity

In this section, we analyze the sensitivity of crucial hyperparameters.

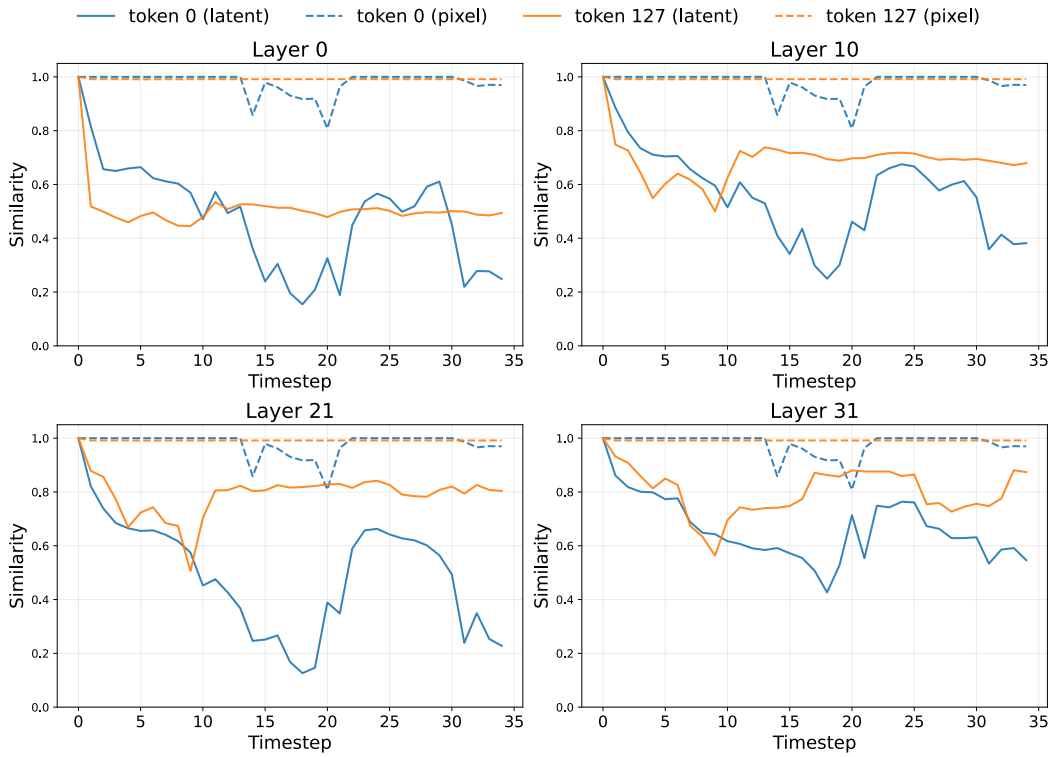


Figure 12: Visually static regions in pixel space exhibit substantial latent representation drift across timesteps.

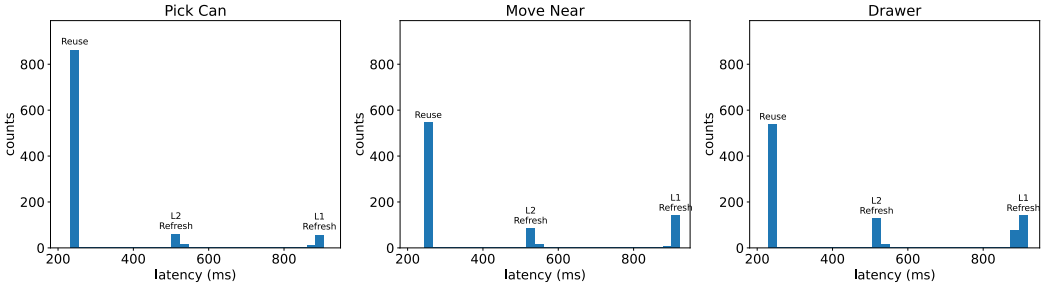


Figure 13: The latency distribution for each step in rollouts. *Reuse* means no refresh needed. *Ln Refresh* means the need to refresh the n-th level static cache.

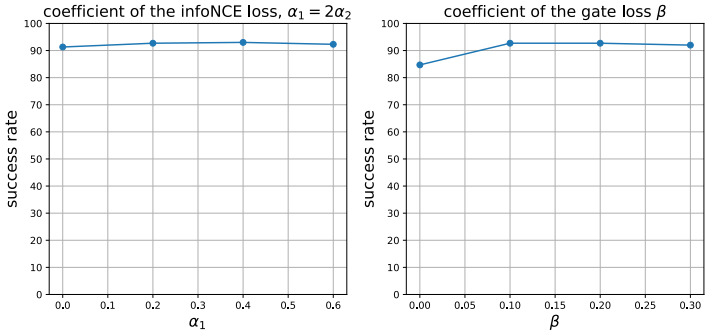


Figure 14: Sensitivity analysis on the coefficients for each loss term. Experiments are conducted on Pick Can (Visual Matching).

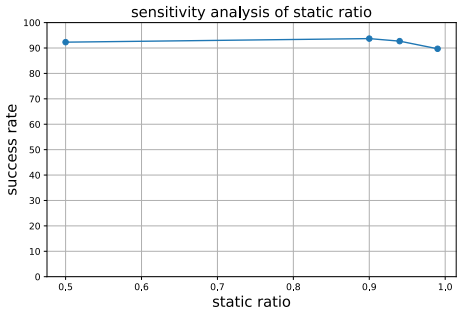
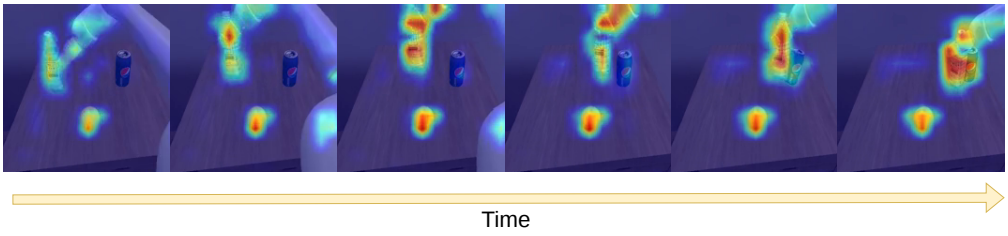


Figure 15: Sensitivity analysis of the static ratio. Experiments are conducted on Pick Can (Visual Matching).



Instruction: Move blue plastic bottle near pepsi can

Figure 16: An illustration that the model misclassifies objects and later corrects such errors.

I.1 Objective Coefficients

We conduct sensitivity analysis of important hyperparameters in our training objective, i.e., α and β in our loss term. The results are plotted in Figure 14.

For the coefficient α of $\mathcal{L}_{\text{infoNCE}}$, we find the performance to be stable across a broad range of values.

For the coefficient β of $\mathcal{L}_{\text{gate}}$, we observe that performance is stable once $\beta \geq 0.1$. When $\beta = 0$, the recache gate is not effectively trained and degenerates to random refresh/reuse decisions, leading to significantly degraded performance.

I.2 Static Ratio

We consider the static ratio as a tunable hyperparameter that can be adapted to different environments. For more dynamic settings, it can be adjusted by (1) reducing the ratio during training to allocate more capacity to dynamic tokens, or (2) lowering the recache threshold for more frequent cache refreshing at test time.

Our sensitivity analysis in Figure 15 shows stable performance across a broad range of ratios, with degradation only at extreme values.

J Temporal Evolution of Static-Dynamic Assignments

Below, we show a representative trajectory illustrating how DySta adapts its static-dynamic assignments over time. As shown in Figure 16, the Pepsi can is initially treated as static because it is not intended to be manipulated. However, when unexpected contact occurs, the model detects the resulting change and reclassifies the can as dynamic, thereby triggering a cache refresh. In some cases, objects such as the orange may be treated as dynamic even though they remain unchanged. This does not affect task correctness, but only introduces a small amount of additional computational overhead.

K Additional Experiments and Scope

We further examine the static-token budget, the allocation between static levels, transfer to an additional base policy, wrist-view observations, and performance on LIBERO-Plus. These experiments complement the acceleration and memory evaluations in the main text. FLOPs percentages are normalized to the unmodified reference policy for each experiment; the reference policy uses 100% FLOPs. The ablations below are reported separately, and comparisons of FLOPs are made within each table.

K.1 Training-matched acceleration baselines

The original main-text acceleration baselines use their training-free protocols. Table 12 additionally continues TTF and VLA-Cache training from the released CogACT checkpoint with the same training configuration as DySta. Their inference-time selection rules remain unchanged. Mean success over the six settings is 75.55% for retrained VLA-Cache and 78.13% for DySta, a 3.4% relative improvement.

Table 12: Training-matched SimplerEnv comparison. VM and VA denote visual matching and variant aggregation. FLOPs are relative to CogACT; latency is in milliseconds.

	VM			VA					
Method	Pick Can	Move Near	Drawer	Pick Can	Move Near	Drawer	FLOPs	Latency	Speedup
CogACT (released)	91.3	85.0	71.8	89.6	80.8	28.3	100%	1360	1.00×
+ TTF (retrained)	89.3	67.1	75.0	85.8	65.0	38.1	86.5%	1051	1.29×
+ VLA-Cache (retrained)	93.0	85.0	73.6	92.3	80.6	28.8	80.1%	985	1.38×
+ DySta	92.7	88.8	75.0	92.4	81.0	38.9	43.4%	601	2.26×

K.2 Multi-frame histories and latency

Table 13 supplements the accuracy results in Table 4. Sampling is at 1 Hz. All latencies use one L40S.

Table 13: History and latency for the LIBERO-Memory results in Table 4. OpenVLA-OFT† uses the largest full-token window we could train.

Method	History (observations)	Latency (ms)
OpenVLA-OFT†	6	1,959
TTF-VLA	20	806
TraceVLA	20	1,888
MemoryVLA	20	778
ContextVLA	20	774
DySta	20	1,171

K.3 Reducing the static-token budget

To test whether the static representation can be made smaller without sacrificing task performance, we halve the static-token budget and retrain the model. Table 14 reports the original and reduced conditions, labeled by their reported total visual-token counts. Halving the static budget improves Pick Can by approximately one percentage point, but reduces Move Near success by 8.75 points under visual matching and 6.25 points under variant aggregation. The reported FLOPs decrease from 35.80% to 29.17% of the base policy. Thus, reducing the static representation offers additional savings but can discard scene information needed for tasks involving multiple objects. We retain the larger budget for the default setting rather than treating the static slots as redundant.

K.4 Allocation between static levels

We hold the total static ratio at 94% and vary its allocation between L1 and L2 (Table 15). A larger L2 allocation gives the highest Pick Can success, whereas the default $0.52/0.42$ allocation gives

Table 14: Separate static-budget ablation on SimplerEnv. The reduced condition halves the static budget and is retrained; the token counts give the total visual-token budgets. VM and VA denote visual matching and variant aggregation. FLOPs are percentages of the unmodified base policy for this ablation.

Metric	DySta (256 tokens)	Half static budget (135 tokens)
VM Pick Can success (%)	92.67	93.67
VA Pick Can success (%)	92.44	93.33
VM Move Near success (%)	88.75	80.00
VA Move Near success (%)	81.04	74.79
FLOPs (% of base policy)	35.80	29.17

the highest Move Near success and the lowest reported FLOPs among the tested splits. Averaged over these two tasks, the $0.12/0.82$ split gains approximately 0.5 percentage points over the default, while requiring approximately 15% more FLOPs. This motivates $0.52/0.42$ as a common efficiency–performance compromise for the default two-level acceleration setting, rather than selecting a different split for each task.

Table 15: Sensitivity to the L1/L2 allocation, with the total static ratio fixed at 0.94. Success rates are percentages. FLOPs are percentages of the unmodified base policy in this separate allocation ablation.

L1 / L2 ratio	Pick Can	Move Near	FLOPs (% of base policy)
$0.12/0.82$	95.33	87.08	45.73
$0.32/0.62$	94.33	84.17	42.74
$0.52/0.42$	92.67	88.75	39.79
$0.72/0.22$	91.00	85.00	43.78
$0.92/0.02$	92.67	86.67	42.41

K.5 An additional base policy: SpatialVLA

We apply DySta to SpatialVLA [37], which uses a PaliGemma2 backbone and a different action tokenizer from the two base policies in the main experiments. We reuse the hyperparameters from the CogACT setting without model-specific retuning and evaluate on SimplerEnv. As shown in Table 16, DySta improves success in four of five settings, with a 0.6-point decrease on VA Move Near, while using 48.6% of the base policy's FLOPs. This extends the evidence to a third base policy with different visual-language and action representations. The table reports computation rather than wall-clock latency, so it does not establish a latency ranking across architectures.

Table 16: SimplerEnv results with SpatialVLA as the base policy. Entries in the five task columns are success rates (%). FLOPs are normalized to unmodified SpatialVLA. VM and VA denote visual matching and variant aggregation.

Model	VM Pick Coke	VM Move Near	VM Open/Close	VA Pick Coke	VA Move Near	FLOPs (%)
SpatialVLA	86.0	77.9	57.4	88.0	72.7	100.0
+ VLA-Cache	81.3	75.8	56.9	85.3	69.4	69.1
+ DySta (Ours)	87.7	79.2	58.3	88.4	72.1	48.6

K.6 Wrist-view observations and the static budget

Camera motion changes the amount of information that can be reused across observations. Table 17 examines two static allocations for wrist-view observations on LIBERO-Goal and includes a third-person operating point for context. With wrist observations, reducing the total static ratio from 0.8 to 0.6 increases success from 95.0% to 97.2%, while increasing the FLOPs fraction from 48.2% to

61.4%. These results illustrate the need for a more conservative reuse budget when the viewpoint changes rapidly.

Table 17: View and static-budget comparison on LIBERO-Goal, reported separately from the default acceleration configuration. FLOPs are percentages of the corresponding unmodified base-policy setting. The third-person row uses a different static allocation and is a reference operating point rather than a matched control for camera view.

Camera view	L1 / L2 ratio	Success (%)	FLOPs (%)
Wrist	$0.3/0.3$	97.2	61.4
Wrist	$0.4/0.4$	95.0	48.2
Third-person	$0.5/0.4$	96.2	47.1

K.7 LIBERO-Plus and the scope of robustness evidence

Table 18 reports results on LIBERO-Plus [10]. Relative to the OpenVLA-OFT† reference configuration, DySta increases Plus-Goal success by 1.3 percentage points and decreases Plus-Long success by 0.3 points, while using 53.4% of the reference FLOPs. These aggregate scores provide additional evidence that the computation savings can be retained in a more challenging evaluation setting.

Table 18: LIBERO-Plus success rates (%). FLOPs are normalized to the OpenVLA-OFT† reference configuration used in this experiment. The dagger identifies the reference configuration for this evaluation.

Model	Plus-Goal	Plus-Long	FLOPs (%)
OpenVLA-OFT†	66.2	54.3	100.0
DySta (Ours)	67.5	54.0	53.4

L Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

M Declaration of LLM Usage

We used large language models solely for editing and polishing the manuscript (e.g., grammar, phrasing, and typo correction). LLMs were not involved in the research ideation, methodology, experiments, or analysis, and do not constitute any original or non-standard component of this work.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The three contributions stated in the introduction are each developed in the Method section and validated in the Experiments section, with scope explicitly stated.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: The paper has included a limitation paragraph in the last section.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: The paper provides computational analysis in the main text and its derivation in the appendix.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The paper describes the experimental setting in the main text and provides the details in the appendix. The code has also been provided in the supplemental material.

   Guidelines:

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

5. **Open access to data and code**

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

Answer: [Yes]

Justification: The code and the instructions have been provided in the supplemental material, and the data is publicly available for the simulation experiments.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper describes the experimental setting in the main text and provides the details in the appendix.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: We do not report error bars across training runs due to the prohibitive computational cost of training VLA models—each reported number can require multiple days on H100-class GPUs—which makes multi-seed training infeasible within the resource budget of this work. This is consistent with standard practice in prior VLA literature. Each result is instead aggregated over multiple evaluation episodes per task to mitigate evaluation variance.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper describes the compute resources used for the experiments in in the appendix.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The paper advances general machine learning research and does not raise any ethical concerns. The authors have reviewed the NeurIPS Code of Ethics and confirm that their research conforms to it.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: The author has included an impact statement section in the appendix.

Guidelines:

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The released assets are a robot manipulation model and a simulation benchmark (LIBERO-Memory), which do not generate harmful content nor involve scraped or sensitive data, and therefore pose no high risk of misuse.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: All pre-trained base models, datasets, and simulation frameworks are properly cited and used in accordance with their original licenses.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: We introduce the benchmark and the codebase. Both are documented in the appendix (dataset construction, BDDL scene description, oracle demonstrations, and implementation details) and provided in the supplemental material.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing. The human-performance reference in the appendix was conducted by two trained volunteers, which only offers as an additional context for understanding the experiments.

    Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve study on human subjects.

Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [Yes]

Justification: The paper includes a declaration in the appendix stating that LLMs were used only for editing and polishing the manuscript, with no involvement in the core methodology.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.