Reflex: Enabling Fast and Predictive Vision-Language-Action Models for Reaction-Critical Manipulation

Yuxuan Chen[1], Wanruo Zhang[1], and Xiao Li[1]

arXiv:2608.14379v1 [cs.RO] 14 Aug 2026

***Abstract*—Vision-Language-Action (VLA) models have recently achieved promising performance in robotic manipulation. However, existing benchmarks mainly evaluate generalization on static manipulation tasks and largely overlook dynamic interaction scenarios. To address this gap, we present *ReflexBench*, a benchmark for reaction-critical manipulation. ReflexBench contains six dynamic tasks and introduces an evaluation framework that decouples simulator stepping from robot control while supporting configurable latency under synchronous and asynchronous inference. Building upon ReflexBench, we propose *ReflexVLA*, an efficient VLA model designed for reaction-critical manipulation without large-scale robot-data pretraining. ReflexVLA enhances temporal reasoning through latent future prediction and multi-frame temporal fusion within the vision backbone, while reducing deployment latency through batched visual encoding and CUDA Graph replay. Experiments show that ReflexVLA consistently improves dynamic manipulation performance while maintaining competitive accuracy on standard static manipulation benchmarks, and real-world experiments further demonstrate its effectiveness under practical deployment conditions. Project website: reflexvla.github.io**

I. Introduction

Recent advances in Vision-Language-Action (VLA) models have significantly expanded the capabilities of robotic manipulation by unifying perception, language understanding, and action generation within a single framework. Building upon the success of large-scale vision-language models, a growing body of work demonstrates that robot policies can be learned from diverse multimodal data and generalized across tasks through natural language instructions. Representative approaches [1] [2] [3] show that scaling model capacity and training data leads to substantial improvements in task diversity, instruction following, and cross-domain generalization. These developments establish VLA models as a promising foundation for general-purpose robot control and motivate further exploration toward increasingly capable embodied agents.

Despite these advances, current VLA models remain challenged in manipulation scenarios that require both rapid reaction and reasoning about future states. Many existing approaches primarily condition actions on the current observation, and their inference latency introduces a delay between perception and execution. This delay becomes particularly consequential in dynamic environments where robots must respond to moving objects, intercept targets at precise times, or manipulate objects whose trajectories affect task success.

To address these limitations, recent research explores several complementary directions. One line of work introduces specialized benchmarks that focus on dynamic manipulation scenarios [4] [5] [6]. In parallel, a growing number of studies investigate how to equip VLA models with capabilities that are better suited for future prediction [4] [7]. Meanwhile, considerable effort is devoted to improving the efficiency of VLA inference through architectural optimization, model compression, and system-level acceleration techniques, with the goal of reducing latency and enabling deployment in real-time robotic applications [8] [9]. However, these research directions are largely explored in isolation. Existing dynamic manipulation benchmarks primarily serve as evaluation platforms without providing a unified framework for studying both anticipation and efficiency, while methods that enhance predictive reasoning often introduce additional computational overhead. Conversely, approaches that prioritize inference acceleration generally focus on efficiency alone and do not explicitly address the need for future-aware decision making. As a result, the challenge of simultaneously achieving strong anticipatory capability and fast inference remains insufficiently explored in current VLA research.

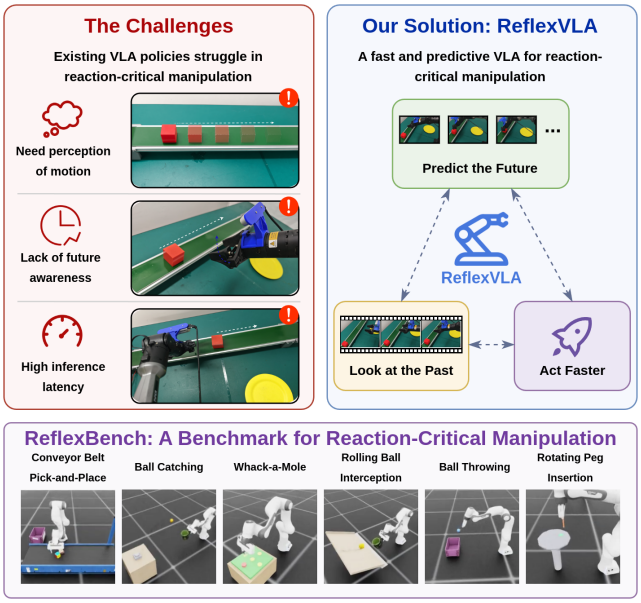

Fig. 1 : Overview of our work. We introduce **ReflexBench**, a benchmark of six reaction-critical manipulation tasks. Building upon it, we propose **ReflexVLA**, a fast and predictive VLA that achieves strong performance with low latency.

In this paper, we first present **ReflexBench**, a benchmark designed to evaluate VLA policies in manipulation tasks that require both efficient execution and anticipation of future environmental dynamics. ReflexBench consists of six simulation-based tasks that capture representative challenges in reaction-critical manipulation. To better reflect practical deployment conditions, the benchmark explicitly incorporates latency effects and aligns the evaluation process as closely as possible

[1] Shanghai Jiao Tong University chen_yuxuan@sjtu.edu.cn

with real-world execution delays. Building upon ReflexBench, we further investigate the key factors that enable VLA policies to succeed in such tasks and propose **ReflexVLA**, a VLA designed for fast and predictive robot control. ReflexVLA is based on three core components. First, it introduces latent prediction of future visual observations, which encourages the policy to reason about near-future environmental states. Second, it leverages multiple historical observations as input, providing richer temporal context for decision making. Third, it incorporates inference latency optimization to reduce the delay between perception and action execution. Extensive experiments demonstrate that these design choices contribute to improved performance, and their combination leads to a more effective solution for dynamic manipulation scenarios that demand both anticipation and efficiency. To summarize our contributions, we

- introduce ReflexBench, a benchmark consisting of six manipulation tasks that require both rapid reaction and future-aware decision making;
- propose ReflexVLA, a fast and predictive VLA framework that combines future latent prediction, multi-frame history modeling, and inference latency optimization;
- conduct extensive experiments in both simulation and real-world environments, demonstrating the effectiveness of the proposed designs.

II. Related Work

A. Benchmarks for Dynamic Manipulation

Recent benchmarks have begun evaluating VLA models in dynamic environments. [4] focuses on object motion and perception-execution latency, while [5] studies spatiotemporal reasoning across 35 dynamic manipulation tasks. [6] further reveals that current VLA models remain highly sensitive to environmental dynamics. In contrast, ReflexBench explicitly models inference latency under both synchronous and asynchronous execution, providing a more faithful evaluation of latency-sensitive robotic manipulation.

B. Efficient Vision-Language-Action Models

As VLA models continue to grow in scale, improving their efficiency becomes increasingly important for real-world deployment. One line of work aims to reduce model size and training cost while maintaining competitive performance, like SmolVLA [8] and VLA-Adapter [10]. Another direction focuses on improving inference efficiency at the system and architecture levels. FASTER [7] revisits the design of flow-based VLAs and explores mechanisms for real-time action generation. VLASH [11] further improves responsiveness through future-state-aware asynchronous inference, reducing the impact of policy latency during execution. In the context of dynamic manipulation, DynamicVLA [4] also considers the challenges introduced by perception-execution delays and investigates VLA architectures that are better suited for dynamic object interaction. Despite these advances, existing efficient VLA methods primarily focus on reducing computational cost or improving execution frequency. Comparatively less attention is devoted to understanding how inference latency interacts with future anticipation in dynamic manipulation.

III. ReflexBench

Recent robotic manipulation benchmarks evaluate VLA models across diverse tasks and environments [12] [13] [14] [15], but largely overlook reaction-critical scenarios. Furthermore, most simulation benchmarks pause the environment during policy inference, ignoring inference latency and the resulting perception-execution delays. Consequently, they fail to accurately assess VLA performance in dynamic real-world settings.

To address these limitations, we introduce ReflexBench, a benchmark specifically designed for reaction-critical manipulation tasks. ReflexBench explicitly incorporates realistic latency effects into the evaluation process and focuses on scenarios that require both future-aware decision making and efficient policy execution.

A. Task List

ReflexBench consists of 6 manipulation tasks that require a policy to infer quickly under dynamic environmental conditions, as shown in Fig. 1 . Concretely, these tasks are:

- **Conveyor Belt Pick-and-Place.** A cube and a distractor object move along a conveyor belt. The robot is required to pick up the cube and place it into a nearby bin.
- **Ball Catching.** The robot must catch an incoming ball whose position changes rapidly over time.
- **Whack-a-Mole.** Targets appear at different locations for a short duration and disappear if not reached in time.
- **Rolling Ball Interception.** A ball rolls down a slope, and the robot must catch it at the edge of the slope to prevent it from falling off.
- **Ball Throwing.** The robot throws a ball toward a bin.
- **Rotating Peg Insertion.** The robot inserts a peg into a continuously rotating socket.

B. Data Collection

Collecting demonstrations for dynamic manipulation tasks is substantially more challenging than for conventional static manipulation benchmarks. In static environments, expert trajectories can often be generated through straightforward scripted policies or teleoperation. In contrast, the tasks in ReflexBench involve moving objects and time-dependent interactions, where successful execution requires continuous adaptation to environmental dynamics.

We employ a planning-based data collection pipeline that decomposes each task into sequential phases with distinct subgoals. For example, in Rotating Peg Insertion, the robot performs pre-alignment, target tracking, and timed insertion. To handle dynamic objects, the planner predicts future trajectories from real-time position and velocity, allowing actions to be planned toward anticipated future states. When planning alone cannot handle complex interaction dynamics, we train a task-specific reinforcement learning policy to generate demonstration trajectories.

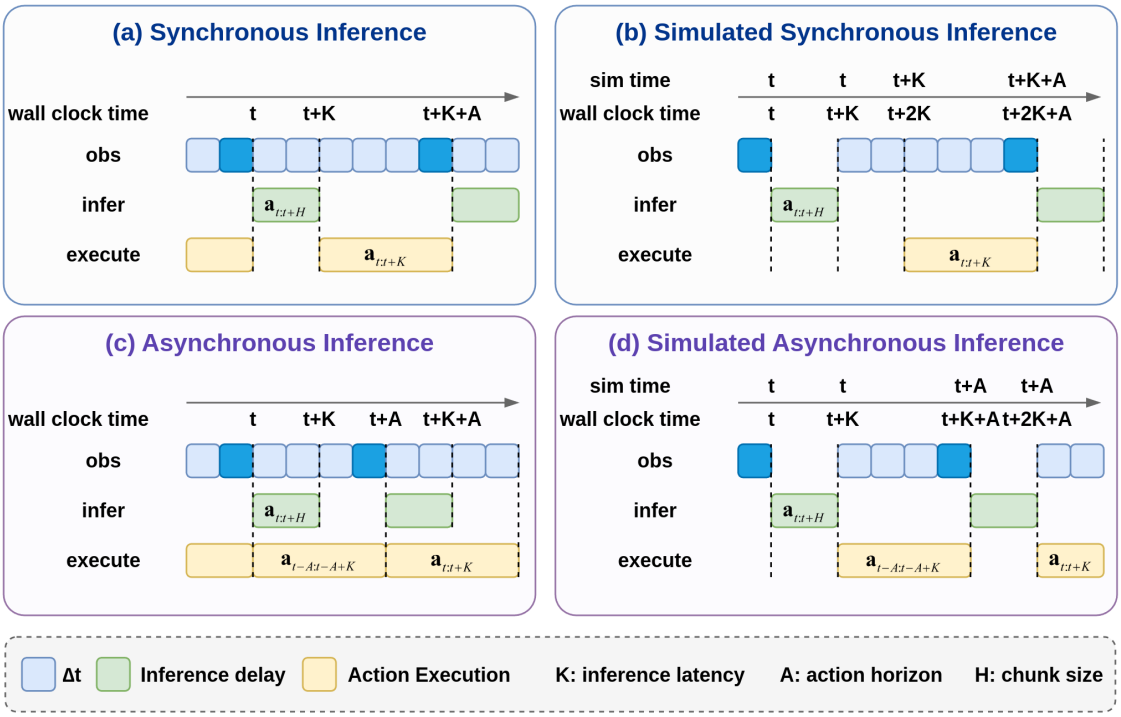

Fig. 2 : Illustration of Four Inference Mechanisms. ReflexBench models the effect of policy inference latency by explicitly simulating the temporal gap between perception and action execution while allowing the latency to be adjusted in a controllable manner.

C. Latency-Aware Evaluation

A key feature of ReflexBench is its latency-aware evaluation protocol. Unlike existing simulation benchmarks that pause the environment during policy inference, ReflexBench decouples simulation from robot control, allowing environmental dynamics to evolve independently of policy computation and enabling more realistic evaluation.

ReflexBench explicitly models inference latency under two common deployment paradigms: synchronous and asynchronous inference. Synchronous inference blocks execution until policy inference completes, whereas asynchronous inference overlaps action execution with policy computation, improving responsiveness at the cost of slightly stale actions.

To faithfully reproduce latency effects in simulation, we introduce a latency blocking mechanism. Fig. 2 illustrates the two inference paradigms and their corresponding simulation protocols. Under synchronous inference, the simulator pauses to collect the current observation and query the policy. After inference, the simulator resumes, the robot remains idle for the specified latency, and then executes the predicted action chunk, faithfully mimicking inference-blocking deployment. Under asynchronous inference, the policy infers a new action chunk from the current observation while the robot executes the action chunk generated in the previous cycle. The newly inferred action becomes available in the next control period.

The latency value can be specified manually, which effectively changes the duration of the blocking window without modifying the policy or hardware configuration. This enables systematic analysis of how different latency levels affect task performance under controlled conditions. In addition, ReflexBench supports latency settings derived from actual policy execution. We first measure the real-world inference latency of a policy and then convert it into the corresponding simulation delay using the Real-Time Factor (RTF), which is defined as

$$\mathrm{RTF} = \frac{t_{\mathrm{sim}}}{t_{\mathrm{wall}}}, \tag{1}$$

where t_{sim} denotes the elapsed simulation time and t_{wall} denotes the corresponding wall-clock time required to execute the simulation. The RTF characterizes the relative speed of the simulator with respect to real time. Given a measured policy inference latency in the real world, we compute the equivalent delay in simulation according to the simulator's RTF and inject the corresponding latency into the evaluation process.

IV. ReflexVLA

ReflexVLA is built on top of a compact vision-language-action architecture [10]. The visual encoder is a fused ViT backbone composed of DINOv2 [16] and SigLIP [17] at 224×224 resolution. The language backbone is a Qwen2.5-0.5B [18] causal transformer. Given a language instruction and multi-view RGB observations, the model inserts projected visual tokens after the beginning-of-sequence token and uses learnable action queries to produce an action chunk through a continuous regression head. As shown in Fig. 3 , ReflexVLA introduces three key designs: latent future prediction for anticipatory representation learning, multi-frame temporal fusion for short-horizon motion awareness, and inference-time computation optimization for responsive deployment.

A. Latent Future Prediction

Dynamic manipulation often requires anticipating future states rather than reacting to the current observation. To

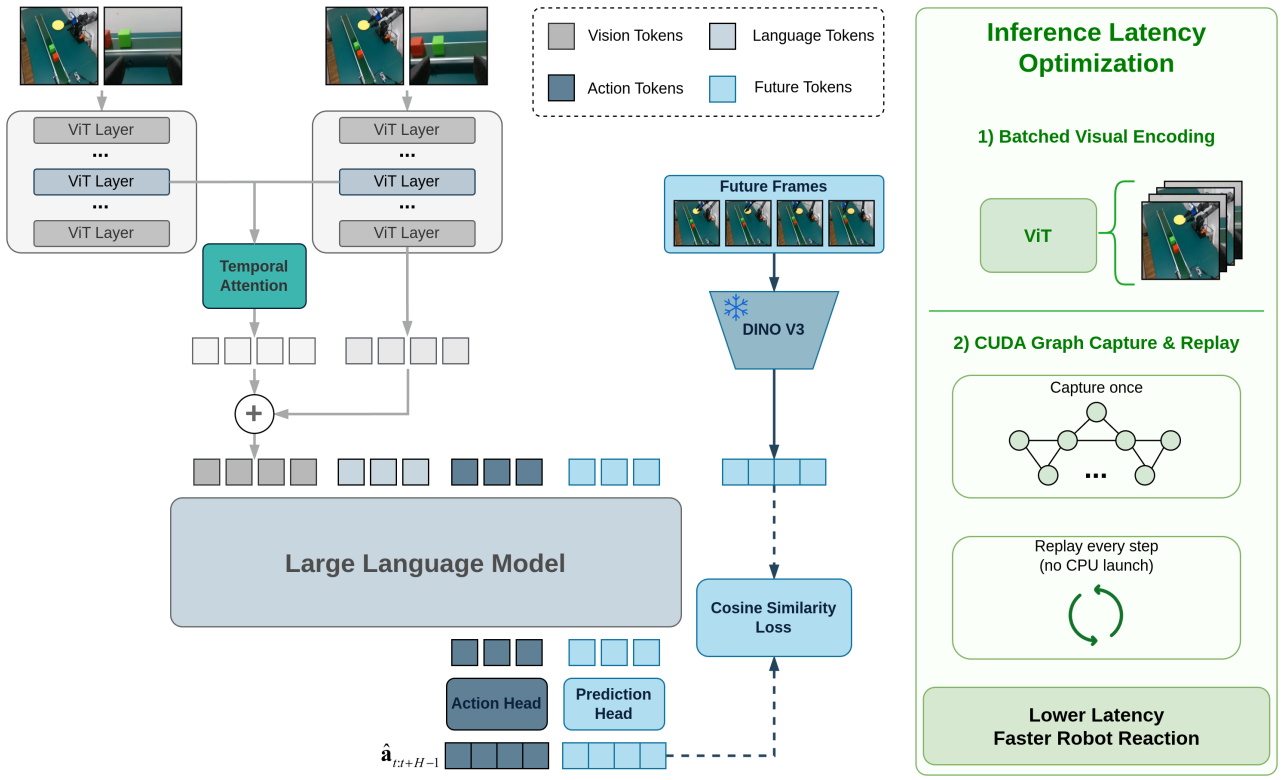

Fig. 3 : Overview of ReflexVLA. ReflexVLA is an efficient vision-language-action model for reaction-critical robotic manipulation, integrating latent future prediction, multi-frame temporal fusion, and inference latency optimization to enable fast and anticipatory decision-making.

this end, ReflexVLA incorporates an auxiliary latent future prediction objective to learn predictive scene representations.

Directly predicting future images is computationally expensive and often forces the model to focus on low-level appearance details that are not directly relevant to control. Instead, we perform future prediction in a semantic latent space extracted by a frozen visual encoder. For each training sample at time step t, the dataset provides a future observation horizon $\{\mathbf{o}_{t+i}\}_{i=1}^{H}$. We encode each future observation using a frozen DINOv3 [19] model:

$$\mathbf{y}_{t+i} = \phi_{\text{DINOv3}}(\mathbf{o}_{t+i}), \qquad \mathbf{y}_{t+i} \in \mathbb{R}^{1024}. \tag{2}$$

To enable future prediction, we append H learnable future tokens to the multimodal input sequence. The prediction horizon H is chosen to be identical to the chunk size used by action prediction, such that each future token corresponds to one future control step within the predicted action chunk. This design aligns future state prediction with the temporal structure of action generation, allowing the model to learn scene dynamics that are directly relevant to downstream control. And each future step is represented by only a single prediction token, making this design significantly efficient. Since the number of prediction tokens is equal to the action chunk size, the additional sequence length remains small in practice. Consequently, the computational and memory overhead introduced by future prediction is negligible relative to the overall cost of the VLA backbone.

After the transformer forward pass, the hidden representation associated with the i-th future token is projected into the DINOv3 feature space:

$$\hat{\mathbf{y}}_{t+i} = f_{\text{pred}}\left(\mathbf{h}_i^{\text{future}}\right), \qquad i = 1, \ldots, H. \tag{3}$$

The predicted feature is trained to match the corresponding future visual representation using a masked cosine similarity loss:

$$\mathcal{L}_{\text{future}} = \frac{\sum_{i=1}^{H} m_i \left[1 - \cos\left(\hat{\mathbf{y}}_{t+i}, \mathbf{y}_{t+i}\right)\right]}{\sum_{i=1}^{H} m_i}, \tag{4}$$

where m_i indicates whether the corresponding future observation is valid.

The future prediction loss is jointly optimized with the action prediction objective:

$$\mathcal{L} = \mathcal{L}_{\text{act}} + \lambda_{\text{future}} \mathcal{L}_{\text{future}}, \tag{5}$$

where $\mathcal{L}_{\text{act}}$ denotes the action learning loss and λ_{future} controls the contribution of future prediction. By requiring the policy to anticipate future visual states during training, ReflexVLA learns representations that capture the temporal evolution of dynamic scenes.

B. Multi-Frame Temporal Fusion

Since motion dynamics cannot be captured from a single observation, the policy requires historical visual context. A straightforward solution is to concatenate multi-frame visual tokens, but this introduces VTP tokens for V views and T frames, with P visual tokens per image, resulting in high computational costs due to quadratic attention complexity.

Moreover, it ignores explicit temporal structures and relies on the language model to learn cross-frame interactions implicitly.

To address these limitations, ReflexVLA performs temporal fusion within the vision backbone and exposes only the fused representation of the current frame to the language model. Given V camera views and T historical observations per view, the vision encoder first extracts patch-level features for all images:

$$\mathbf{X}^{\text{mid}}, \mathbf{X}^{\text{final}} \in \mathbb{R}^{B\times V\times T\times P\times D}, \tag{6}$$

where $\mathbf{X}^{\text{mid}}$ and $\mathbf{X}^{\text{final}}$ denote intermediate-layer and final-layer visual features, respectively.

We perform temporal fusion on intermediate features, which retain richer local appearance and motion cues than deeper, more semantic representations.

For each camera view v and patch location p, features from the same spatial position across the T historical frames form a temporal trajectory $\mathbf{x}^{\text{mid}}_{v,p,1:T}$. The features are first normalized and projected into a lower-dimensional space:

$$\mathbf{z}_{v,p,1:T} = \text{Down}\left(\text{LN}\left(\mathbf{x}^{\text{mid}}_{v,p,1:T}\right)\right) + \mathbf{e}_{1:T}, \tag{7}$$

where $\mathbf{e}_{1:T}$ denotes temporal positional embeddings and $\text{Down}(\cdot)$ reduces the feature dimension to improve efficiency.

A causal temporal attention layer is then applied:

$$\Delta\mathbf{x}_{v,p,t} = \text{Up}\left(\text{MHA}_{\text{causal}}\left(\mathbf{z}_{v,p,1:T}\right)_T\right). \tag{8}$$

The causal mask prevents information leakage from future observations and ensures consistency with online deployment. The resulting feature summarizes the relevant motion history for the current patch.

The resulting temporal representation is integrated into the current-frame visual feature:

$$\tilde{\mathbf{x}}_{v,p,t} = \mathbf{x}^{\text{final}}_{v,p,t} + \Delta\mathbf{x}_{v,p,t}. \tag{9}$$

All fused current-frame tokens are then flattened as

$$\tilde{\mathbf{X}}_t \in \mathbb{R}^{B\times VP\times D}, \tag{10}$$

projected into the language model embedding space, and inserted into the multimodal sequence for action prediction.

An important advantage of this design is that the number of visual tokens consumed by the language model remains identical to that of the single-frame setting. Consequently, ReflexVLA can exploit short-term motion information without introducing the substantial language-model-side overhead associated with multi-frame images.

C. Inference Latency Optimization

Inference latency is a critical factor in dynamic manipulation, where even strong policies can fail if actions are generated too slowly. To improve real-world responsiveness, ReflexVLA reduces end-to-end inference latency by optimizing two common sources of overhead in VLA systems: visual processing and GPU execution.

The first optimization targets visual encoding. Since ReflexVLA consumes multiple camera views and historical observations, a naive implementation would independently process each image through the vision backbone, resulting in $V \times T$ separate forward passes. Such repeated execution introduces unnecessary framework overhead and leads to suboptimal GPU utilization. Instead, ReflexVLA batches all view-frame images into a single visual encoder invocation.

A single batched forward pass then produces visual features for all observations simultaneously, after which the resulting features are reshaped into $\mathbb{R}^{B\times V\times T\times P\times D}$ for temporal fusion. This formulation naturally aligns with the multi-frame architecture of ReflexVLA while significantly reducing per-frame execution overhead.

The second optimization focuses on GPU execution efficiency. During deployment, the computation graph of ReflexVLA remains fixed, including visual encoding, temporal fusion, multimodal projection, language model inference, and action prediction [9]. Rather than repeatedly launching individual GPU kernels for every control cycle, we capture the complete inference pipeline as a CUDA Graph:

$$\hat{\mathbf{a}}_{t:t+H-1} = \mathcal{G}\left(\mathbf{x}_{\text{text}}, \mathbf{x}_{\text{vision}}, \mathbf{x}_{\text{state}}\right), \tag{11}$$

where $\mathcal{G}$ denotes the captured CUDA Graph and $\hat{\mathbf{a}}_{t:t+H-1}$ represents the predicted action chunk. During execution, new observations are copied into pre-allocated buffers and the graph is replayed directly, eliminating repeated kernel scheduling and runtime dispatch overhead.

Notably, these optimizations do not modify the policy architecture or learning objective. Instead, they reduce deployment overhead while preserving the original multimodal reasoning process. The system-level optimizations substantially improve execution efficiency and make ReflexVLA more suitable for latency-sensitive robotic manipulation tasks.

V. Experiments

In our experiments, we focus on the following questions:

- Q1: How do different factors influence the success rate of reaction-critical manipulation tasks?
- Q2: Does ReflexVLA achieve better performance on ReflexBench than existing VLA baselines?
- Q3: Does ReflexVLA still achieve competitive performance on standard static manipulation tasks?
- Q4: What is the contribution of each design component to the overall performance of ReflexVLA?
- Q5: How well does ReflexVLA perform in real-world manipulation tasks?

A. Setup

Baselines. We compare ReflexVLA with a diverse set of representative VLA models. Specifically, we include the lightweight models VLA-Adapter [10] and SmolVLA [8], the dynamic manipulation methods DynamicVLA [4] and PUMA [5], and the large-scale VLA models OpenVLA-OFT [20] and $\pi_{0.5}$ [21].

Metrics. We evaluate task performance using *task success rate* and report *model size* as an indicator of deployment cost. We additionally measure end-to-end *inference latency* in the latency ablation.

Implementation Details. Unless otherwise specified, all VLA policies are trained as a single policy on the same

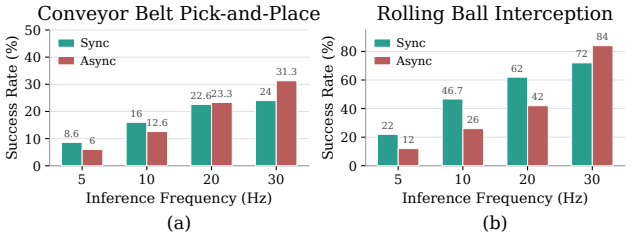

Fig. 4 : Success rates of SmolVLA under synchronous and asynchronous inference across different inference frequencies on two tasks. Chunk size is set to 8.

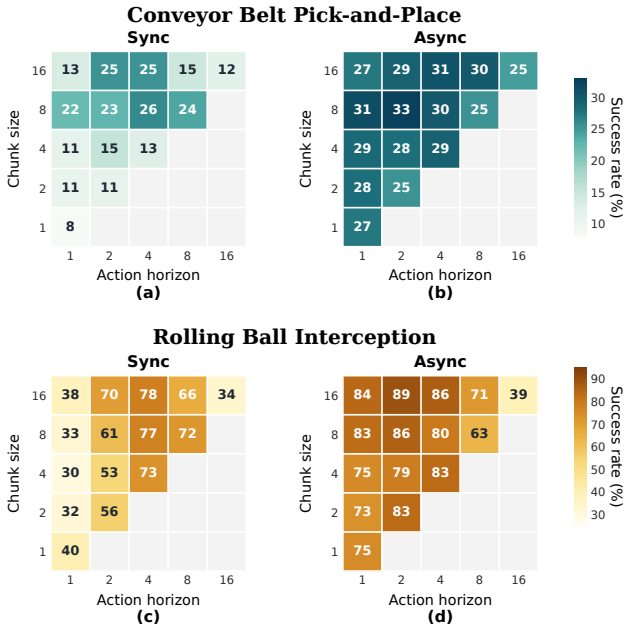

Fig. 5 : Success rate matrices of SmolVLA under different chunk sizes and action horizons on two tasks. Chunk size is the policy's training hyperparameter, while action horizon is the number of action steps executed during inference. The inference frequency is fixed at 30 Hz.

dataset, consisting of 200 demonstration episodes for each of the 6 tasks. ReflexVLA is trained with 2 consecutive temporal observations as visual input and a latent future prediction weight of $\lambda_{\text{future}} = 0.05$. All policies use asynchronous inference with an action chunk size of 8 and an action horizon of 2, and are evaluated on a single NVIDIA RTX 5880 Ada GPU. Each success rate is computed over 150 evaluation episodes per task and reported as the mean and standard deviation across three runs with different random seeds.

B. Main Results

The best performance is achieved by combining a larger chunk size with a shorter action horizon under high-frequency asynchronous inference (Q1). Before evaluating different VLA policies, we first investigate how the inference paradigm, inference frequency, action chunk size, and action horizon affect policy performance on reaction-critical manipulation tasks. Here, the chunk size specifies the number of actions predicted per chunk, while the action horizon determines how many actions are executed before the next inference. To enable efficient exploration, we conduct this study using the lightweight SmolVLA model on two representative ReflexBench tasks, namely Conveyor Belt Pick-and-Place and Rolling Ball Interception.

Fig. 4 compares synchronous and asynchronous inference across different inference frequencies. At low frequencies, asynchronous inference suffers from greater observation-action mismatch, which is particularly detrimental in rapidly changing environments. As the inference frequency increases, this mismatch is reduced, enabling asynchronous inference to increasingly benefit from continuous action execution and eventually outperform synchronous inference.

Fig. 5 studies the effect of chunk size and action horizon at a fixed inference frequency of 30 Hz. Larger chunk sizes with shorter action horizons under asynchronous inference generally achieve better performance, although the optimal chunk size varies slightly across tasks. We therefore use asynchronous inference with a chunk size of 8 and an action horizon of 2 for all subsequent experiments on ReflexBench to ensure fair comparisons.

ReflexVLA achieves competitive average success rates on ReflexBench while maintaining a relatively compact model size (Q2). Table I demonstrates that ReflexVLA consistently achieves competitive performance across the six tasks in ReflexBench and obtains the highest overall average success rate among all evaluated methods. Specifically, ReflexVLA achieves an average success rate of 50.4%, outperforming all existing baselines while using only a 1B-parameter model. Compared with its backbone model, VLA-Adapter, ReflexVLA improves the average success rate from 30.3% to 50.4%, demonstrating the effectiveness of the proposed designs. Notably, ReflexVLA surpasses substantially larger general VLA models, including OpenVLA-OFT (7B) and $\pi_{0.5}$ (4B), and achieves comparable overall performance to the dynamic-oriented PUMA while requiring only one-quarter of its model size.

ReflexVLA maintains competitive performance on standard static manipulation benchmarks while being primarily designed for dynamic manipulation scenarios (Q3). Table II reports the results on the LIBERO benchmark [12]. Although ReflexVLA is specifically designed to improve reaction-critical dynamic manipulation, it achieves an average success rate of 97.2%, which is comparable to the best-performing VLAs. Compared with its backbone model, VLA-Adapter, ReflexVLA exhibits only a marginal difference in overall performance, indicating that the proposed future prediction and temporal modeling modules do not compromise the model's capability on conventional static manipulation tasks. These results demonstrate that the improvements introduced by ReflexVLA enhance dynamic manipulation performance while preserving strong generalization to standard robotic manipulation benchmarks.

C. Ablation Study

Latent future prediction enhances anticipatory reasoning, multi-frame temporal fusion improves motion understanding, and inference latency optimization reduces

TABLE I: Performance comparison on ReflexBench. Results are reported as mean success rate (%) ± standard deviation over three runs.

Model	Params	Conveyor Belt Pick-and-Place	Ball Catching	Whack-a-Mole	Rolling Ball Interception	Ball Throwing	Rotating Peg Insertion	Avg.
OpenVLA-OFT [20]	7B	58.0±1.2	5.3±1.2	**100.0±0.0**	41.4±4.2	10.0±3.3	1.3±0.7	36.0
$\pi_{0.5}$ [21]	4B	39.1±1.0	6.0±0.7	98.9±0.3	36.8±1.8	34.0±2.7	6.7±1.4	36.9
PUMA [5]	4B	67.4±1.7	4.0±0.7	**100.0±0.0**	**85.1±3.4**	33.8±1.0	11.1±1.7	50.2
DynamicVLA [4]	0.5B	30.6±2.4	4.4±1.0	90.2±1.7	45.8±1.7	**36.2±4.7**	16.7±4.2	37.3
SmolVLA [8]	0.5B	19.3±2.0	2.9±0.8	**100.0±0.0**	70.2±2.0	27.1±1.0	10.0±4.6	38.3
VLA-Adapter [10] (Baseline)	1B	36.8±2.4	6.0±2.0	68.4±3.9	23.1±7.7	29.1±5.2	**18.4±2.5**	30.3
ReflexVLA (Ours)	**1B**	**73.8±4.1**	**7.3±0.7**	**100.0±0.0**	77.1±7.1	31.7±1.0	12.4±1.5	**50.4**

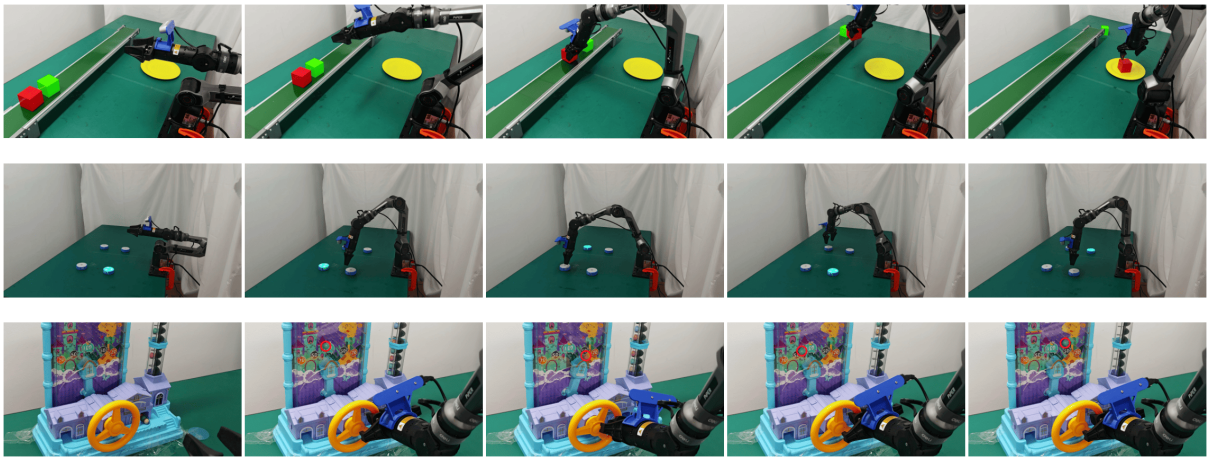

Fig. 6 : Real-world rollout examples. From top to bottom: Conveyor Belt Pick-and-Place, PressButtons, and CatchBalls.

TABLE II: Results on LIBERO. Accuracy is reported in (%).

Model	Spatial	Object	Goal	Long	Avg.
OpenVLA-OFT [20]	97.6	98.4	97.9	94.5	97.1
$\pi_{0.5}$ [21]	**98.8**	98.2	**98.0**	92.4	96.9
VLA-Adapter [10]	97.8	**99.2**	97.2	**95.0**	**97.3**
ReflexVLA (Ours)	98.2	**99.2**	**98.0**	93.6	97.2

TABLE III: Progressive ablation study of ReflexVLA.

Method	Variant	SR (%)	Lat. (ms)
Baseline	–	36.8	81.522
+ Future Pred.	Trainable	4.9 (-31.9)	82.508
	Frozen	62.8 (+26.0)	82.508
+ Temporal Fusion	Cross Attn.	66.1 (+29.3)	127.335
	MHA	68.2 (+31.4)	125.824
	MHA (Middle)	71.7 (+34.9)	125.107
+ Latency Opt.	Batch Enc. + CUDA Graph	**73.8** (+37.0)	**64.991**

deployment delay, together leading to higher success rate with lower inference latency (Q4). Table III presents a progressive ablation study of ReflexVLA, where each proposed component is introduced incrementally starting from the baseline model.

We first investigate the latent future prediction module. Directly training the model to predict future visual representations with a trainable visual target results in a substantial performance degradation, indicating that jointly optimizing the representation space and the prediction objective leads to unstable supervision. In contrast, using features extracted from a frozen DINOv3 encoder provides a stable semantic prediction target and improves the success rate from 36.8% to 62.8% with almost no additional inference overhead.

Building upon this model, we further evaluate three temporal fusion strategies. Notably, performing temporal fusion on intermediate visual features instead of the final-layer representations yields the best performance of 71.7%. This observation suggests that intermediate features preserve richer motion-related information, making them more suitable for modeling temporal dynamics than highly semantic final-layer features.

Finally, we incorporate the proposed inference latency optimization, including batched visual encoding and CUDA Graph replay. These optimizations further improve the success rate to 73.8% while simultaneously reducing the inference latency from 125.1 ms to 65.0 ms. The improved success rate demonstrates that reducing deployment latency is particularly important for reaction-critical manipulation tasks.

D. Real-World Experiments

Real-World Setup. We validate ReflexVLA through real-world experiments on an AgileX Piper robotic arm. Three representative tasks are chosen: Conveyor Belt Pick-and-Place, PressButtons, and CatchBalls. Performance is measured by task-specific metrics: success rate for Conveyor Belt Pick-and-Place, correctly pressed buttons within 30 seconds for

TABLE IV: Quantitative results of the real-world experiments. Conveyor Belt reports the number of successful trials out of 20 attempts. PressButtons reports the average number of buttons pressed within 30 seconds, and CatchBalls reports the average number of balls caught out of 10 balls. Each policy is evaluated 20 times on every task.

Model	Conveyor	PressButtons	CatchBalls
SmolVLA [8]	2/20	0.9	3.8
PUMA [5]	13/20	20.8	5.4
Ours	**16/20**	**22.5**	**6.7**

PressButtons, and successful catches out of 10 attempts for CatchBalls. We compare ReflexVLA with two representative baselines, SmolVLA [8] and PUMA [5], with all policies trained using teleoperation demonstrations.

ReflexVLA demonstrates strong real-world manipulation performance (Q5). Table IV presents the quantitative results, and Figure 6 provides representative execution examples. Compared with SmolVLA, ReflexVLA consistently achieves higher performance across all three tasks, indicating that its temporal modeling and latency-aware designs are critical to these tasks. Despite using a much smaller model than PUMA, ReflexVLA achieves comparable overall performance on these dynamic manipulation tasks, suggesting that modeling future scene evolution together with efficient temporal perception is an effective approach for real-world reactive manipulation.

E. Limitations

Our work has several limitations that warrant future investigation. First, although ReflexVLA introduces latent future prediction and multi-frame temporal fusion, these components are incorporated only during fine-tuning rather than through large-scale pretraining, which may limit their full potential. Second, our study focuses on two widely adopted inference paradigms, naive synchronous and asynchronous inference. More advanced inference mechanisms, such as RTC [22], are not explored in this work and may provide additional improvements in latency-sensitive manipulation scenarios.

VI. Conclusion

In this work, we present ReflexBench, a benchmark for latency-aware dynamic manipulation. ReflexBench contains six representative dynamic tasks and introduces a latency-aware evaluation framework that decouples simulator stepping from robot control. We also present ReflexVLA, a latency-aware vision-language-action framework for dynamic robotic manipulation. By integrating latent future prediction, multi-frame temporal fusion, and inference optimization, ReflexVLA enables robots to better anticipate future states while achieving efficient real-time execution. Experiments on ReflexBench demonstrate that ReflexVLA achieves competitive performance among existing VLA baselines while maintaining the efficiency of a lightweight 1B parameter model. Furthermore, ReflexVLA remains competitive on the standard static manipulation benchmark LIBERO. Real-world experiments on three reaction-critical manipulation tasks further validate the effectiveness and practicality of our approach.

References

[1] A. Brohan, N. Brown, J. Carbajal et al., "Rt-2: Vision-language-action models transfer web knowledge to robotic control," arXiv preprint arXiv:2307.15818, 2023.

[2] M. J. Kim, K. Pertsch, S. Karamcheti et al., "Openvla: An open-source vision-language-action model," arXiv preprint arXiv:2406.09246, 2024.

[3] K. Black, N. Brown, D. Driess et al., "π0: A vision-language-action flow model for general robot control, 2024," arXiv preprint arXiv:2410.24164, 2024.

[4] H. Xie, B. Wen, J. Zheng, Z. Chen, F. Hong, H. Diao, and Z. Liu, "Dynamicvla: A vision-language-action model for dynamic object manipulation," arXiv preprint arXiv:2601.22153, 2026.

[5] H. Fang, S. Li, S. Wang, X. Xi, D. Liang, and X. Bai, "Towards generalizable robotic manipulation in dynamic environments," arXiv preprint arXiv:2603.15620, 2026.

[6] Y. Zhang, C. Song, V. Singh, X. Li, K. Ye, Z. Hu, Z. Pu, Y. Yin, and V. Chaudhary, "Overcoming dynamics-blindness: Training-free pace-and-path correction for vla models," arXiv preprint arXiv:2605.11459, 2026.

[7] Y. Lu, Z. Liu, X. Fan, Z. Yang, J. Hou, J. Li, K. Ding, and H. Zhao, "Faster: Rethinking real-time flow vlas," arXiv preprint arXiv:2603.19199, 2026.

[8] M. Shukor, D. Aubakirova, F. Capuano et al., "Smolvla: A vision-language-action model for affordable and efficient robotics," arXiv preprint arXiv:2506.01844, 2025.

[9] Y. Ma, Y. Zhou, Y. Yang, T. Wang, and H. Fan, "Running vlas at real-time speed," arXiv preprint arXiv:2510.26742, 2025.

[10] Y. Wang, P. Ding, L. Li, C. Cui, Z. Ge, X. Tong, W. Song, H. Zhao, W. Zhao, P. Hou et al., "Vla-adapter: An effective paradigm for tiny-scale vision-language-action model," in Proceedings of the AAAI conference on artificial intelligence, vol. 40, no. 22, 2026, pp. 18 638–18 646.

[11] J. Tang, Y. Sun, Y. Zhao, S. Yang, Y. Lin, Z. Zhang, J. Hou, Y. Lu, Z. Liu, and S. Han, "Vlash: Real-time vlas via future-state-aware asynchronous inference," arXiv preprint arXiv:2512.01031, 2025.

[12] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "Libero: Benchmarking knowledge transfer for lifelong robot learning," in Advances in Neural Information Processing Systems, vol. 36, 2023, pp. 44 776–44 791.

[13] S. Tao, F. Xiang, A. Shukla et al., "Maniskill3: Gpu parallelized robotics simulation and rendering for generalizable embodied ai," arXiv preprint arXiv:2410.00425, 2024.

[14] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in Conference on robot learning. PMLR, 2020, pp. 1094–1100.

[15] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun et al., "Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation," in Conference on Robot Learning. PMLR, 2023, pp. 80–93.

[16] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby et al., "Dinov2: Learning robust visual features without supervision," Transactions on Machine Learning Research Journal, 2024.

[17] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," in Proceedings of the IEEE/CVF international conference on computer vision, 2023, pp. 11 975–11 986.

[18] A. Yang, B. Yang, B. Zhang et al., "Qwen2.5 technical report," arXiv preprint arXiv:2412.15115, 2024.

[19] O. Siméoni, H. V. Vo, M. Seitzer, F. Baldassarre, M. Oquab, C. Jose, V. Khalidov, M. Szafraniec, S. Yi, M. Ramamonjisoa et al., "Dinov3," arXiv preprint arXiv:2508.10104, 2025.

[20] M. J. Kim, C. Finn, and P. Liang, "Fine-tuning vision-language-action models: Optimizing speed and success," arXiv preprint arXiv:2502.19645, 2025.

[21] K. Black, N. Brown et al., "$\pi_{0.5}$: a vision-language-action model with open-world generalization," in Proceedings of The 9th Conference on Robot Learning, vol. 305, 27–30 Sep 2025, pp. 17–40.

[22] K. Black, M. Galliker, and S. Levine, "Real-time execution of action chunking flow policies," in Advances in Neural Information Processing Systems, vol. 38, 2025, pp. 33 383–33 407.