TURBOVLA: REAL-TIME VISION-LANGUAGE-ACTION MODEL AT 32 HZ ON A CONSUMER-GRADE GPU WITH <1 GB VRAM

Anonymous authors
Paper under double-blind review

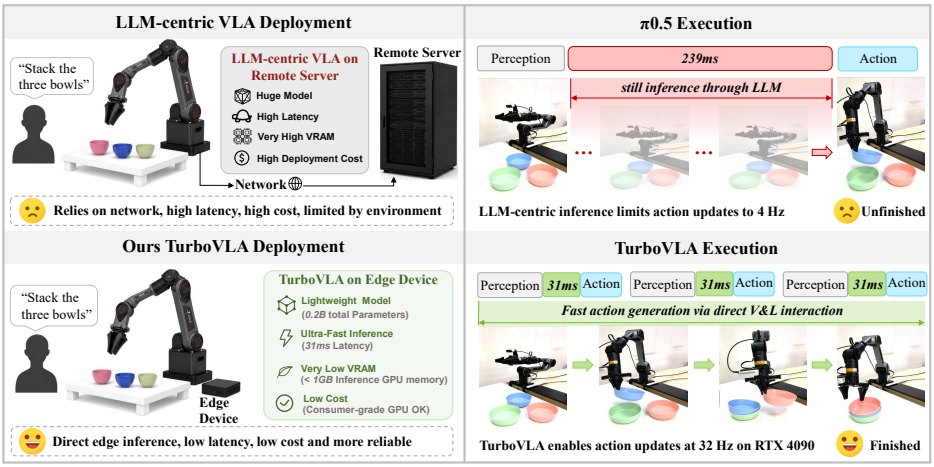


Figure 1: TurboVLA enables compact local deployment and real-time language-conditioned manipulation with 0.2B parameters, 0.9 GiB inference VRAM and 31.2 ms policy latency.

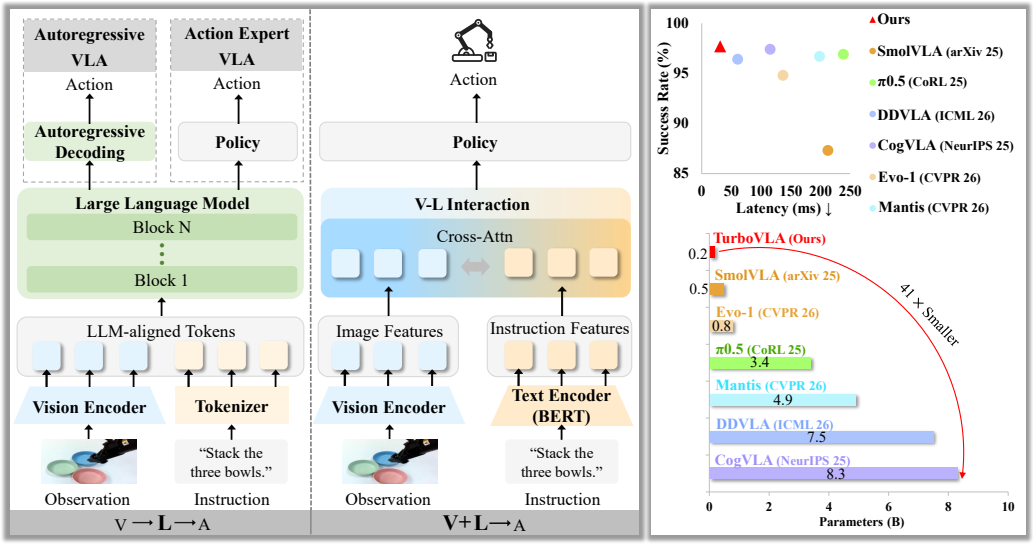
ABSTRACT

Vision-language-action (VLA) models commonly adopt an LLM-centric $V \rightarrow \boldsymbol{L} \rightarrow A$ pathway, processing visual observations and language instructions through a large language model before predicting robot actions. Although effective, this design incurs substantial computation and memory overhead. In this work, we introduce **TurboVLA**, a compact VLA architecture built on a direct $\boldsymbol{V} + \boldsymbol{L} \rightarrow A$ mapping. Instead of using a large language model as the central interface between perception and action, TurboVLA independently encodes visual observations and language instructions, directly exchanges information between them through lightweight bidirectional vision-language interaction, and predicts continuous action chunks with a compact decoder. This simple design directly constructs task-conditioned representations while avoiding the overhead of an LLM-centric execution pathway. On LIBERO, TurboVLA achieves 97.6% average success with only 0.2B parameters, 31.2 ms inference latency, and 0.9 GiB inference VRAM on a consumer-grade RTX 4090. Notably, a 0.4 B TurboVLA achieves 88.06% success on RoboTwin 2.0, even matching or outperforming substantially larger VLA policies. These results demonstrate that the simple $\boldsymbol{V} + \boldsymbol{L} \rightarrow A$ design of TurboVLA can achieve high performance without requiring an LLM-centric execution pathway, offering a new perspective on how vision, language, and action can be connected for efficient robotic manipulation. Code will be available.

1 INTRODUCTION

Vision-language-action (VLA) models have become a powerful framework for language-conditioned robotic manipulation, connecting visual observations, language instructions, and robot actions within a unified policy (Brohan et al., 2023; Octo Model Team et al., 2024; Black et al., 2025b;a; Liu et al., 2025; Li et al., 2024; Fu et al., 2025; Fang et al., 2026). A common design is to place a large language model at the center of this process. Such systems effectively follow an indirect

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



(a) The comparison between existing LLM-centric VLA (left) and our method (right) (b) Performance Improvement

Figure 2: (a) LLM-centric VLA predicts actions from large-language-model representations, whereas TurboVLA directly fuses visual and instruction features for continuous control. (b) TurboVLA achieves highly competitive LIBERO performance with substantially lower latency and model scale.

$V \rightarrow \boldsymbol{L} \rightarrow A$ pathway: visual observations and language instructions are mapped into LLM-aligned tokens, which are subsequently decoded into robot actions (Driess et al., 2023; Zitkovich et al., 2023; Kim et al., 2024). This design transfers broad semantic knowledge from large-scale pretraining to robot control and supports semantic generalization and high-level reasoning.

LLM-centric VLA models, however, can impose a substantial computational bottleneck on real-time robotic execution, where low latency is essential for responsive interaction, high-throughput manipulation, and deployment on resource-constrained robotic platforms. As summarized in Fig. 2(a), these models typically generate actions either autoregressively as action tokens (Kim et al., 2024; Zitkovich et al., 2023) or through action experts that predict continuous action chunks (Black et al., 2025b;a; Kim et al., 2025; Li et al., 2024; Shukor et al., 2025). Although action-expert designs avoid the token-by-token decoding overhead of autoregressive language generation, they still rely on a large language or multimodal backbone to process visual observations and instructions. The resulting computational and memory overhead increases inference latency and limits control frequency. This raises a more fundamental question: *how to design a simple, elegant and efficient VLA that directly maps vision and language to actions for execution-level manipulation, without centering on a large language model?*

Our key observation is that execution-level control need not be centered on a large language model. When an instruction specifies the intended manipulation skill, the execution policy does not need to perform explicit high-level reasoning or subtask generation. Instead, the policy can primarily focus on grounding the instruction in the current visual observations and using this information to guide action. In LLM-centric VLA models, this interaction is mediated through a general-purpose language-model representation. We argue that a lightweight text encoder, such as BERT (Devlin et al., 2019), can capture execution-relevant instruction semantics, while compact cross-modal interaction allows visual and linguistic features to inform each other (Lynch & Sermanet, 2020; Shridhar et al., 2022; Jang et al., 2022; Mees et al., 2022b). This motivates a direct vision-language interaction design that constructs control-oriented representations for continuous action prediction.

Therefore, we propose **TurboVLA**, a simple yet efficient $\boldsymbol{V} + \boldsymbol{L} \rightarrow A$ model for real-time language-conditioned manipulation. As shown in Fig. 2(a), TurboVLA separately encodes visual observations and task instructions using a visual encoder and a lightweight text encoder. Inspired by the efficient cross-modal interaction in Grounding DINO (Liu et al., 2024), TurboVLA uses a compact cross-attention module to fuse visual and linguistic features and then decodes the resulting representation into continuous action chunks (Zhao et al., 2023). By directly connecting visual and linguistic features to action prediction, TurboVLA avoids the overhead of an LLM-centric execution pathway, significantly reducing inference latency and GPU memory usage.

Extensive experiments show that TurboVLA offers a favorable performance–efficiency trade-off. On LIBERO (Liu et al., 2023), TurboVLA achieves 97.6% average success, with 0.2B parameters, 31.2 ms policy inference latency and <1 GB of inference VRAM on a consumer-grade RTX 4090. On RoboTwin 2.0 (Chen et al., 2026), a 0.4B TurboVLA variant achieves 88.06% average success, even matching or outperforming substantially larger LLM-centric VLA systems. Real-world experiments further validate the effectiveness and efficiency of TurboVLA. TurboVLA substantially reduces the computational and memory requirements for deploying language-conditioned manipulation policies in latency-sensitive and resource-constrained robotic systems. These results suggest that lightweight language encoding and direct vision-language interaction enable competitive and efficient execution-level manipulation without an LLM-centric policy. Our contributions are summarized as follows:

- We revisit the computational and memory costs of LLM-centric VLA models and highlight how lightweight instruction encoding and direct bidirectional vision-language interaction can retain task-relevant language semantics for execution-level control, without placing a large language model between perception and action.
- We propose TurboVLA, a simple and efficient $\boldsymbol{V} + \boldsymbol{L} \rightarrow A$ model that separately encodes visual observations and task instructions, and allows the two modalities to jointly construct task-conditioned, control-oriented representations through direct bidirectional vision-language interaction for continuous action prediction.
- We validate TurboVLA on simulation benchmarks and real-world manipulation tasks, demonstrating competitive manipulation performance with reduced computational and memory requirements. These results highlight the potential for more accessible robotic deployment and provide insight into efficient policy design beyond LLM-centric architectures.

2 Related Work

Vision-language-action models. Vision-language-action (VLA) models map visual observations and task instructions to robot actions, often leveraging large-scale vision-language pretraining for semantic generalization. RT-1 (Brohan et al., 2023) demonstrates scalable transformer-based control, while RT-2 (Zitkovich et al., 2023) and OpenVLA (Kim et al., 2024) adapt pretrained vision-language models through action tokens. Continuous-control models such as π_0 (Black et al., 2025b) and $\pi_{0.5}$ (Black et al., 2025a) attach dedicated action experts to pretrained multimodal backbones. Other generalist policies explore reusable policy representations (Octo Model Team et al., 2024), diffusion-based action generation (Liu et al., 2025; Li et al., 2024), and foundation models for diverse embodiments (Bjorck et al., 2025; Bu et al., 2025; Wang et al., 2026d). Recent advances further enrich policy representations through visual foresight (Yang et al., 2026b), predictive world knowledge (Zhang et al., 2025), and latent reasoning (Bai et al., 2026). TurboVLA focuses on the architecture used for execution-level control: in contrast to LLM-centric policies, it separately encodes task text and visual observations, directly fuses their features, and predicts actions conditioned on robot state without a large generative language backbone.

Efficient execution in VLA policies. Existing approaches improve VLA efficiency at several levels. Action-side methods improve action generation through parallel decoding or action-chunk prediction (Kim et al., 2025; Liang et al., 2026b; Liu et al., 2025), as well as compact action tokenization (Pertsch et al., 2025; Liu et al., 2026). Compact architectures such as TinyVLA (Wen et al., 2025), SmolVLA (Shukor et al., 2025), and Evo-1 (Lin et al., 2026) reduce model scale or inference cost. A complementary line of work reduces backbone inference cost through quantization (Xu et al., 2026), token reuse and pruning (Xu et al., 2025; Jiang et al., 2025), adaptive depth and structural pruning (Yang et al., 2025; Wang et al., 2026a), and distillation (Jeon et al., 2026). Other approaches improve inference or execution efficiency through asynchronous action-chunk execution (Black et al., 2025c), streaming inference with horizon-aware flow sampling (Lu et al., 2026), and speculative inference (Niu et al., 2026). These methods optimize computation or execution within existing policy frameworks. TurboVLA instead targets the representation pathway, replacing the large generative language backbone with compact encoders and direct vision-language interaction for control.

Language interfaces for robot control. Natural-language instructions have long been used as task specifications for robotic manipulation. Early imitation-learning methods explored language grounding for instruction-conditioned robotic control (Lynch & Sermanet, 2020; Stepputtis et al.,

2020). CLIPort (Shridhar et al., 2022) and BC-Z (Jang et al., 2022) further explored language-conditioned manipulation, while CALVIN (Mees et al., 2022b) provides a benchmark for long-horizon language-conditioned control and HULC (Mees et al., 2022a) develops policies for this setting. TurboVLA investigates whether compact instruction encoding and direct vision-language interaction can match the performance of an LLM-centric representational pathway while retaining efficiency in the execution-level VLA setting.

3 TURBOVLA

We introduce **TurboVLA**, a simple and elegant $\boldsymbol{V} + \boldsymbol{L} \rightarrow A$ model for execution-level language-conditioned manipulation. As illustrated in Fig. 3(a), TurboVLA separately encodes observations and instructions, allows the two modalities to exchange information through bidirectional cross-modal interaction, and combines the resulting representation with the robot state to predict continuous action chunks. This design preserves task-relevant semantics through a lightweight language encoder without placing a large language model as the central interface between perception and action, enabling competitive manipulation performance with efficient inference.

3.1 PROBLEM FORMULATION AND APPROACH OVERVIEW

At execution step t, a language-conditioned manipulation policy receives multi-view visual observations $\mathcal{O}_t$, a task instruction x, and the current robot state s_t, and predicts a chunk of H continuous actions $\hat{\mathbf{A}}_t = \pi_\theta(\mathcal{O}_t, x, s_t) \in \mathbb{R}^{H \times d_a}$, where d_a is the action dimension. The predicted actions are executed sequentially before the policy observes the next state and predicts a new action chunk.

LLM-centric VLA implementation. Existing LLM-centric VLA models process visual observations and task instructions through a pretrained large language model:

$$H_t^L = F_L([P_v(E_v(\mathcal{O}_t)) ; \mathrm{Tok}(x)]) , \tag{1}$$

where E_v is the visual encoder, P_v projects visual features into the language-model embedding space, $\mathrm{Tok}(x)$ embeds the instruction, and F_L denotes the large language model. The resulting multimodal representation supports either autoregressive action-token prediction (Kim et al., 2024) or continuous action generation through an action module (Black et al., 2025b;a). We summarize this pathway as $V \rightarrow \boldsymbol{L} \rightarrow A$, where $\boldsymbol{L}$ denotes an LLM-centric multimodal interface that goes beyond text encoding to integrate visual and linguistic information into representations for action prediction. Thus, both action generation designs retain the language-model backbone as the representational bridge between perception and action.

Our approach. Unlike LLM-centric VLA, TurboVLA separately encodes visual observations and task instructions and integrates their features through direct bidirectional vision-language interaction. Specifically, the instruction, visual observations, and robot state are encoded as $Z^l = \Phi_l(x)$, $Z_t^v = \Phi_v(\mathcal{O}_t)$, and $Z_t^s = \Phi_s(s_t)$, respectively, where Φ_l is a lightweight instruction encoder. The resulting vision-language representation is combined with the robot-state features to predict an action chunk:

$$Z_t^{vl} = \Phi_{vl}(Z_t^v, Z^l), \quad \hat{\mathbf{A}}_t = D_\theta\big(Q_a, [Z_t^{vl}; Z_t^s]\big) . \tag{2}$$

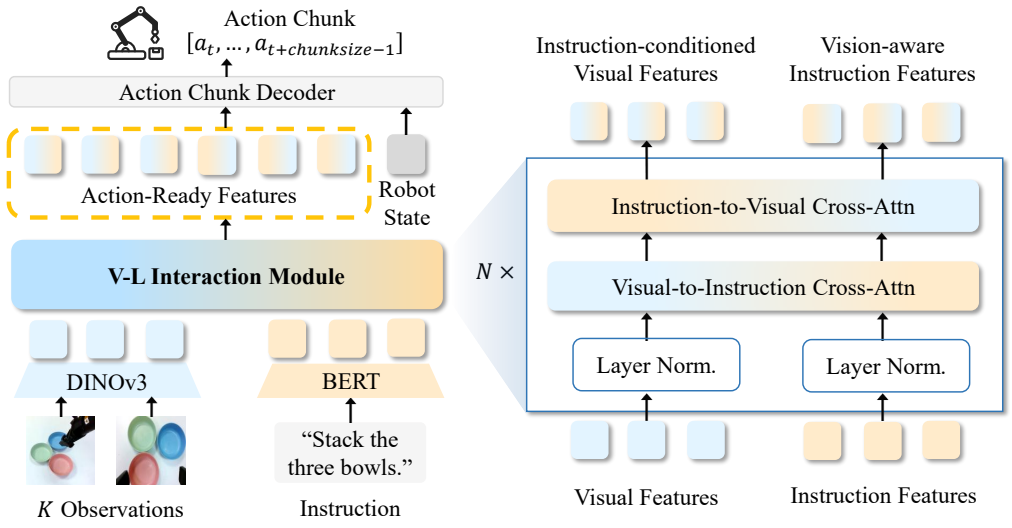
Here, Φ_l, Φ_v, and Φ_s denote the instruction, visual, and robot state encoders, respectively, Φ_{vl} denotes the direct vision-language interaction module, and D_θ denotes the action decoder.

3.2 MULTIMODAL FEATURE ENCODING

Lightweight instruction encoding. To reduce the overhead of an LLM-centric execution pathway while retaining simple yet sufficient instruction understanding, TurboVLA separately encodes task instructions and visual observations, as shown in Fig. 3(a). Given a task instruction x, a lightweight text encoder such as BERT (Devlin et al., 2019) extracts token-level instruction features:

$$Z^l = P_l\left(f_{\text{text}}(x)\right) \in \mathbb{R}^{N_l \times d}, \tag{3}$$

where P_l projects the encoder output into the hidden dimension d and N_l is the number of instruction tokens. This lightweight encoding provides a compact semantic representation of the task instruction.



(a) Overall Architecture (b) Vision-Language Interaction Module

Figure 3: Overview of TurboVLA. (a) TurboVLA simply yet efficiently maps visual observations and language instructions to continuous action chunks through compact modality encoders, vision-language interaction, and an action chunk decoder. (b) The interaction module is designed to be as simple as possible. It uses stacked bidirectional cross-attention to produce vision-aware instruction features and instruction-conditioned visual features.

Multi-view visual encoding. For each camera observation $I_t^{(i)}$, the image encoder extracts spatial visual features, which are projected and augmented with positional and camera-view embeddings:

$$Z_t^{v,(i)} = P_v\left(f_{\text{img}}\left(I_t^{(i)}\right)\right) + E_{\text{pos}}^{(i)} + e_{\text{view}}^{(i)}, \qquad Z_t^v = \left[Z_t^{v,(1)}; \ldots; Z_t^{v,(K)}\right]. \tag{4}$$

Here, $E_{\text{pos}}^{(i)}$ preserves within-view spatial structure and $e_{\text{view}}^{(i)}$ identifies the camera source. Concatenating the K streams retains complementary cues from multiple viewpoints.

Robot state encoding. We encode the current robot state s_t separately, where f_{state} is a lightweight projection network and N_s is the number of state tokens:

$$Z_t^s = f_{\text{state}}(s_t) \in \mathbb{R}^{N_s \times d}, \tag{5}$$

These state features are fed directly to the action decoder to condition action prediction on the robot's current configuration.

3.3 Vision-Language Interaction Module

Independently encoded visual and textual features do not yet identify which visual content is relevant to the current instruction. While LLM-centric VLAs perform this alignment within a large language backbone, TurboVLA instead uses the simple and efficient vision-language interaction module in Fig. 3(b) to directly exchange information between the two streams.

Let $V_t^0 = Z_t^v$ and $L_t^0 = Z^l$ denote the initial visual and instruction features. The interaction module progressively updates both streams through N bidirectional cross-modal layers:

$$\left(V_t^\ell, L_t^\ell\right) = \text{FusionLayer}_\ell\left(V_t^{\ell-1}, L_t^{\ell-1}\right), \qquad \ell = 1, \ldots, N. \tag{6}$$

Within each fusion layer, the two streams exchange information through bidirectional cross-attention. We omit the layer index on these projections for notational simplicity:

$$V_t^\ell = V_t^{\ell-1} + \text{Attn}_{v \leftarrow l}\left(V_t^{\ell-1}, L_t^{\ell-1}\right), \qquad L_t^\ell = L_t^{\ell-1} + \text{Attn}_{l \leftarrow v}\left(L_t^{\ell-1}, V_t^{\ell-1}\right). \tag{7}$$

Visual-to-instruction attention conditions visual features on task semantics, while instruction-to-visual attention injects scene context into the instruction stream. After the final layer, the updated streams are concatenated as

$$Z_t^{vl} = \left[V_t^N; L_t^N\right]. \tag{8}$$

Through this compact interaction module, information including target objects, attributes, and spatial relations can modulate the relevant visual features, while the instruction representation is simultaneously adapted to the current scene. This simple interaction design efficiently provides task-specific multimodal information for action prediction without relying on the broad generative and reasoning capacity of a large language model.

3.4 Continuous Action Chunk Prediction

We use a lightweight ACT-style transformer decoder (Zhao et al., 2023) to map the fused multimodal representation and robot state features to a sequence of continuous actions:

$$\hat{\mathbf{A}}_t = D_\theta\big(Q_a, \big[Z_t^{\mathrm{vl}}; Z_t^s\big]\big) \in \mathbb{R}^{H\times d_a}, \tag{9}$$

where $Q_a = [q_1, \ldots, q_H]$ contains H learnable action queries, H is the prediction horizon, d_a is the action dimension, and D_θ denotes the action decoder. We train TurboVLA through behavior cloning with an ℓ_1 loss between the predicted chunk $\hat{\mathbf{A}}_t$ and the expert sequence $\mathbf{A}_t^* = [a_t^*, \ldots, a_{t+H-1}^*]$.

4 Experiments

We evaluate whether TurboVLA can retain strong language-conditioned manipulation performance while substantially reducing the model scale, inference latency, and memory overhead of LLM-centric VLA policies. Experiments cover single-arm manipulation on LIBERO (Liu et al., 2023), bimanual manipulation on RoboTwin 2.0 (Chen et al., 2026), and real-world deployment, followed by ablations of language conditioning and vision-language interaction.

4.1 Experimental Setup

We use DINOv3 (Siméoni et al., 2025) as the visual backbone and BERT (Devlin et al., 2019) as the instruction encoder. Visual and linguistic features are projected to a dimension $d = 256$ and processed by $N = 6$ bidirectional interaction layers initialized from pretrained Grounding DINO weights (Liu et al., 2024). We use an ACT-style decoder (Zhao et al., 2023) to predict continuous action chunks from the resulting representations. Training details are provided in Appendix A.

LIBERO contains four suites, each comprising ten language-conditioned manipulation tasks. We use the modified `no_noops` RLDS datasets released with OpenVLA (Kim et al., 2024) and jointly train one mixed-suite model with a DINOv3 ViT-B backbone. The model is trained for 40k steps with a batch size of 128. Following the VLA-Adapter rollout protocol (Wang et al., 2026c), we conduct 50 rollouts per task and report suite-level and average success rates over 2,000 trials.

RoboTwin 2.0 comprises 50 language-conditioned bimanual manipulation tasks requiring coordinated dual-arm control. Following prior work (Bi et al., 2026; Yang et al., 2026a; Wang et al., 2026b; Wu et al., 2026a), we jointly train TurboVLA with a DINOv3 ViT-L backbone on 2,500 clean and 25,000 randomized demonstrations. Training lasts 150k steps with a batch size of 512. Following the StarVLA (StarVLA Community, 2026) evaluation framework, we conduct 100 rollouts per task in both clean and randomized settings.

Real-world evaluation. We conducted real-world experiments using an AgileX Piper platform illustrated in Fig. 4. We consider four representative language-conditioned manipulation tasks: *grab roller*, *move playing card away*, *press stapler*, and *stack three bowls*. Together, these tasks involve accurate object grounding, multi-object manipulation and precise interaction. We initialize the policy from the TurboVLA pretrained on LIBERO and fine-tune it on 4×65 teleoperated real-world demonstrations for 12.5k steps. Each task is evaluated over 40 trials. We compare against $\pi_{0.5}$ under the same platform, fine-tuning data, and evaluation protocol.

4.2 Main Results

Tab. 1 and Tab. 2 present complementary evaluations of TurboVLA on simulation benchmarks. From these results, we draw the following observations.

1) Lightweight instruction encoding and direct vision-language interaction enable competitive and efficient manipulation. As summarized in Tab. 1, TurboVLA achieves 97.6% average success

Table 1: Comparison on LIBERO. "Emb. PT." denotes additional embodied pretraining on robot data beyond LIBERO. Params denotes total parameters. Latency denotes time from multimodal input to producing an action chunk or an equivalent number of autoregressive action tokens. Both latency and Inference VRAM are measured on a single RTX 4090 with batch size one. For TurboVLA, the reported parameter count corresponds to the DINOv3 ViT-B configuration.

Method	Reference	Emb. PT.	Deployment Efficiency			LIBERO Success Rate (%)				
			Params (B)↓	VRAM (GiB)↓	Latency (ms)↓	Spa.	Obj.	Goal	Long	Avg.↑
Non-VLA policy baseline										
Diffusion Policy (Chi et al., 2023)	RSS 23	✗	0.3	1.1	924.8	78.3	92.5	68.3	50.5	72.4
Capability-oriented VLAs										
OpenVLA (Kim et al., 2024)	CoRL 24	✓	7.5	14.9	202.9	84.7	88.4	79.2	53.7	76.5
π_0 (Black et al., 2025b)	RSS 25	✓	3.2	7.2	219.2	96.8	98.8	95.8	85.2	94.2
UniVLA (Bu et al., 2025)	RSS 25	✓	7.6	15.0	173.8	96.5	96.8	95.6	92.0	95.2
$\pi_{0.5}$ (Black et al., 2025a)	CoRL 25	✓	3.4	7.7	238.8	98.8	98.2	98.0	92.4	96.9
CogVLA (Li et al., 2025)	NeurIPS 25	✓	8.3	16.1	115.5	98.6	98.8	96.6	95.4	97.4
Mantis (Yang et al., 2026b)	CVPR 26	✓	4.9	7.9	198.7	98.8	99.2	94.4	94.2	96.7
MM-ACT (Liang et al., 2026a)	CVPR 26	✗	8.2	16.3	723.2	97.8	99.4	94.8	93.0	96.3
VLA-JEPA (Sun et al., 2026)	ECCV 26	✓	2.8	5.3	108.7	96.2	99.6	97.2	95.8	97.2
VEGA-3D (Wu et al., 2026b)	ECCV 26	✓	9.0	16.0	546.4	97.4	99.4	97.0	95.2	97.3
Acceleration-oriented VLAs										
OpenVLA-OFT (Kim et al., 2025)	RSS 25	✓	7.7	15.7	112.2	97.6	98.4	97.9	94.5	97.1
DDVLA (Liang et al., 2026b)	ICML 26	✓	7.5	14.5	60.8	97.2	99.4	96.8	92.2	96.4
Lightweight VLAs										
SmolVLA (Shukor et al., 2025)	ArXiv 25	✗	0.5	1.4	212.7	90.0	96.0	92.0	71.0	87.3
DreamVLA (Zhang et al., 2025)	NeurIPS 25	✗	0.7	1.5	128.0	97.5	94.0	89.5	89.5	92.6
VLA-Adapter (Wang et al., 2026c)	AAAI 26	✗	1.5	4.3	87.3	97.8	99.2	97.2	95.0	97.3
Evo-1 (Lin et al., 2026)	CVPR 26	✗	0.8	1.7	137.2	92.7	97.7	96.3	92.3	94.8
TurboVLA (Ours)	-	✗	**0.2**	**0.9**	**31.2**	97.0	100.0	98.2	95.2	**97.6**

Table 2: Comparison on RoboTwin 2.0. "Emb. PT." denotes additional embodied pretraining on robot data beyond RoboTwin 2.0, and Params denotes total parameters. "Clean" and "Randomized" denote the success rates under the clean and randomized evaluation settings, respectively. For TurboVLA, the reported parameter count corresponds to the DINOv3 ViT-L configuration.

Method	Reference	Emb. PT.	Deployment Efficiency		RoboTwin 2.0 Success Rate (%)		
			Params (B)↓	Latency (ms)↓	Clean	Rand.	Avg.↑
π_0 (Black et al., 2025b)	RSS 25	✓	3.2	214.5	65.92	58.40	62.16
$\pi_{0.5}$ (Black et al., 2025a)	CoRL 25	✓	3.4	246.5	82.74	76.76	79.75
Motus (Bi et al., 2026)	CVPR 26	✓	8.7	2070.4	88.66	87.02	87.84
X-VLA (Zheng et al., 2026)	ICLR 26	✓	0.9	87.3	72.80	72.84	72.82
GigaWorld-Policy (Ye et al., 2026)	ArXiv 26	✓	12.4	654.2	86.36	85.04	85.70
ABot-M0 (Yang et al., 2026a)	ArXiv 26	✓	5.9	172.6	86.06	85.08	85.57
Qwen-VLA (Wang et al., 2026b)	ArXiv 26	✓	–	–	86.10	87.20	86.65
LingBot-VLA (Wu et al., 2026a)	ArXiv 26	✓	4.2	164.4	88.56	86.68	87.62
TurboVLA (Ours)	-	✗	**0.4**	**43.4**	88.02	88.10	**88.06**

with 0.2B parameters, 31.2 ms inference latency, and 0.9 GiB inference VRAM on LIBERO (Liu et al., 2023). Compared with capability-oriented VLAs, it achieves higher average success than $\pi_{0.5}$ at 96.9% while using approximately 6% of its parameters and providing $8\times$ faster inference. The favorable trade-off also holds against acceleration-oriented and lightweight VLAs. OpenVLA-OFT (Kim et al., 2025) and DDVLA (Liang et al., 2026b) achieve inference latencies of 112.2 ms and 60.8 ms, respectively, while TurboVLA achieves both higher average success and lower latency without dedicated acceleration mechanisms. Among lightweight VLAs, TurboVLA exceeds the performance of Evo-1 (Lin et al., 2026) and SmolVLA (Shukor et al., 2025) with fewer parameters and faster inference. Together, these comparisons suggest that lightweight instruction encoding can provide task-relevant semantics for direct vision-language interaction, supporting competitive execution-level manipulation without the computational and memory costs of an LLM-centric policy.

2) This simple and lightweight design extends to challenging bimanual multi-task control. Beyond single-arm manipulation, we evaluate whether TurboVLA can support coordinated bimanual control on RoboTwin 2.0 (Chen et al., 2026). Recent models such as Qwen-VLA (Wang et al., 2026b) and LingBot-VLA (Wu et al., 2026a) achieve strong performance by benefiting from large-scale

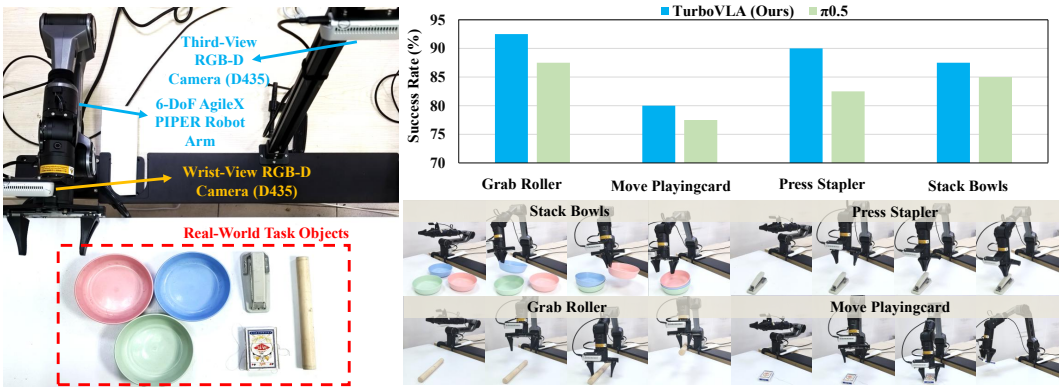


Figure 4: Real-world evaluation on the AgileX Piper platform. Left: our single-arm setup with a wrist-view RGB-D camera and a third-view RGB-D camera, together with the objects used in the four tasks. Right top: success-rate comparison between TurboVLA and $\pi_{0.5}$ on four real-world manipulation tasks. Right bottom: qualitative execution examples of TurboVLA.

embodied pretraining. Nevertheless, TurboVLA achieves a higher average success rate than both models, reaching 88.06% with only 0.4B parameters and 43.4 ms inference latency as shown in Tab. 2. It also matches the performance of Motus (Bi et al., 2026) and ABot-M0 (Yang et al., 2026a), despite their substantially larger model sizes. Compared with LingBot-VLA, TurboVLA reduces inference latency from 164.4 ms to 43.4 ms, providing approximately 3.8× faster inference. It further outperforms $\pi_{0.5}$ (Black et al., 2025a) while providing approximately 5.7× faster inference. These results show that the simple TurboVLA design remains highly effective for challenging bimanual manipulation with substantially less computation and no embodied pretraining.

3) Architectural efficiency translates into practical deployability. Compared with LLM-centric policies such as $\pi_{0.5}$, TurboVLA substantially reduces model size, inference latency, and GPU memory requirements, as summarized in Tab. 1. This efficiency is accompanied by competitive real-world performance: on the AgileX Piper platform, TurboVLA achieves 87.5% average success across four manipulation tasks, performing comparably to $\pi_{0.5}$, as shown in Fig. 4. These results demonstrate that TurboVLA combines competitive physical manipulation with low inference cost, supporting its suitability for real-world execution-level control.

4.3 Ablation Study

We conduct ablations on LIBERO to examine the role of semantic language conditioning, the choice of instruction encoder, and the design of vision-language interaction. Additional ablations on LIBERO and RoboTwin 2.0 are reported in Appendix C.1.

Semantic language conditioning. We first study the role of language itself. Tab. 3 shows that removing language reduces the average success rate from 97.6% to 68.5%, with the largest drop on LIBERO-Goal. This degradation indicates that visual observations alone are insufficient when the same scene can correspond to different instructed goals. Learned task-ID embeddings recover average success to 96.6%, but remain 1.0% below semantic instructions. This gap suggests that semantic instruction features provide more effective conditioning than task identity alone.

Instruction encoder. Then, we examine whether the proposed architecture depends on a specific text backbone. As shown in Tab. 4, under otherwise identical settings, T5-Base (Raffel et al., 2020) achieves a competitive 97.6% average success rate, while the SigLIP (Zhai et al., 2023) text encoder reaches 97.3%. These results support the effectiveness of lightweight instruction encoding for execution-level control and show that TurboVLA is robust to the choice of text backbone.

Vision-language interaction design. Having established the importance of semantic instruction features, we next study how visual and language features should interact before action decoding. As illustrated in Fig. 5, we compare no interaction, two asymmetric cross-attention variants, and the proposed bidirectional interaction, while keeping all other architecture and training settings unchanged. As shown in Tab. 5, direct concatenation achieves 96.3% average success, while the two one-way cross-attention variants improve it to 96.8% and 97.2%. These gains suggest that explicit

Table 3: Effect of language conditioning.

Condition	Spa.	Obj.	Goal	Long	Avg.
w/o Language	87.4	99.8	8.6	78.0	68.5
Task-ID Embedding	96.8	99.6	97.2	92.6	96.6
Semantic Instruction	97.0	100.0	98.2	95.2	**97.6**

Table 4: Effect of the instruction encoder.

Text Encoder	Overall Params (M)	Spa.	Obj.	Goal	Long	Avg.
SigLIP-Base	216.9	98.4	98.8	96.0	95.8	97.3
T5-Base	216.2	98.8	99.4	97.0	95.2	**97.6**
BERT	216.1	97.0	100.0	98.2	95.2	**97.6**

Table 5: Effect of vision-language interaction.

Interaction Design	Spa.	Obj.	Goal	Long	Avg.
w/o Interaction	98.8	99.8	95.6	90.8	96.3
Language Queries Visual	96.8	99.6	97.0	93.8	96.8
Visual Queries Language	96.8	99.6	98.6	93.6	97.2
Bidirectional Interaction	97.0	100.0	98.2	95.2	**97.6**

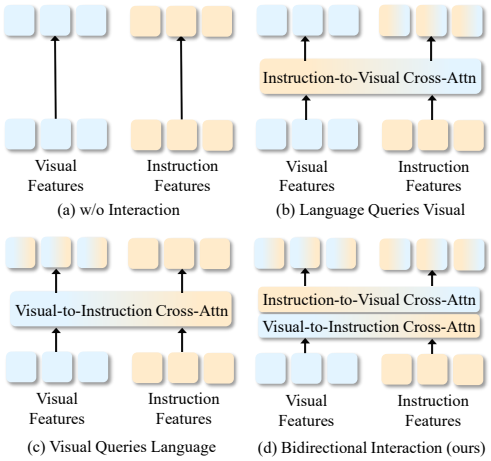


Figure 5: Vision-language interaction variants.

interaction before action decoding is beneficial even when the decoder already has access to both modalities. Bidirectional interaction performs best at 97.6%, indicating that allowing visual and instruction features to condition each other benefits action prediction.

Overall, these ablations support the proposed direct $\boldsymbol{V} + \boldsymbol{L} \rightarrow A$ pathway, showing that lightweight semantic instruction encoding and bidirectional vision-language interaction together enable effective execution-level control without an LLM-centric policy.

4.4 Runtime Acceleration of TurboVLA

Table 6: Runtime acceleration of TurboVLA.

Method	No Graph	CUDA Graph
TurboVLA	31.2 ms	11.3 ms
TurboVLA + Opt.	**30.3 ms**	**7.42 ms**

Building on the low inference latency of TurboVLA, we further integrate CUDA Graph replay with the policy and apply several runtime optimizations (Opt.). As shown in Tab. 6, the resulting implementation remarkably reduces the policy inference latency to 7.42 ms on an RTX 4090. These refinements target data conversion, memory allocation, and unused intermediate outputs, while preserving the full visual and language encoders and the original action horizon. This latency corresponds to a policy inference rate above 130 Hz, suggesting that policy inference itself is unlikely to be the bottleneck for real-time manipulation. In this regime, the practical control frequency is more likely to be limited by sensing, communication, and actuator dynamics than by the execution policy, further demonstrating the deployment potential of TurboVLA in latency-sensitive robotic systems.

5 Conclusion

In this paper, we propose TurboVLA, a simple and efficient $\boldsymbol{V} + \boldsymbol{L} \rightarrow A$ model that combines lightweight instruction encoding with direct bidirectional vision-language interaction for execution-level manipulation. Experiments on LIBERO, RoboTwin 2.0, and real-world manipulation tasks show that TurboVLA preserves task-conditioned manipulation capability while significantly reducing model size, inference latency, and memory consumption. Our results suggest that execution-level control does not necessarily require a general-purpose LLM as the central interface between perception and action, offering insight into how vision and language can be organized for efficient robotic control.

Limitations and future work. Nevertheless, TurboVLA currently focuses on concrete execution-level manipulations and is not specifically designed for open-ended commands such as “clean the room”, where the robot needs to rely on high-level reasoning and task decomposition to identify relevant subgoals and organize their execution. Future work will explore extending TurboVLA to high-level and long-horizon tasks, aiming to achieve more intelligent manipulation while retaining its efficient execution pathway. It will also broaden evaluation beyond simulation benchmarks and tabletop scenarios to more diverse real-world settings, including cluttered household environments, office environments, industrial workspaces, and other robot embodiments.

6 AI USE STATEMENT

In this work, we have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, assisting with translation, cleaning and reformatting dataset, supporting qualitative and thematic data analysis, interpreting results. We used generative AI tools for improving the readability of the manuscript and creating software code. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Shuanghao Bai, Jing Lyu, Wanqi Zhou, Zhe Li, Dakai Wang, Lei Xing, Xiaoguang Zhao, Pengwei Wang, Zhongyuan Wang, Cheng Chi, et al. Latent reasoning vla: Latent thinking and prediction for vision-language-action models. In *Proc. of Intl. Conf. on Machine Learning*, 2026.

Hongzhe Bi, Hengkai Tan, Shenghao Xie, Zeyuan Wang, Shuhe Huang, Haitian Liu, Ruowen Zhao, Yao Feng, Chendong Xiang, Yinze Rong, et al. Motus: A unified latent action world model. In *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pp. 35101–35113, 2026.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Proc. of the Conference on Robot Learning*, volume 305, pp. 17–40. PMLR, 27–30 Sep 2025a.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. In *Proc. of Robotics: Science and Systems*, 2025b.

Kevin Black, Manuel Galliker, and Sergey Levine. Real-time execution of action chunking flow policies. In *Proc. of Advances in Neural Information Processing Systems*, volume 38, pp. 33383–33407, 2025c.

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. In *Proc. of Robotics: Science and Systems*, 2023.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. In *Proc. of Robotics: Science and Systems*, 2025.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. In *Proc. of Intl. Conf. on Machine Learning*, 2026.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proc. of Robotics: Science and Systems*, 2023.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. In *Proc. of Intl. Conf. on Machine Learning*, 2023.

Heng Fang, Shangru Li, Shuhan Wang, Xuanyang Xi, Dingkang Liang, and Xiang Bai. Towards generalizable robotic manipulation in dynamic environments. In *Proc. of European Conference on Computer Vision*, 2026.

Haoyu Fu, Diankun Zhang, Zongchuang Zhao, Jianfeng Cui, Dingkang Liang, Chong Zhang, Dingyuan Zhang, Hongwei Xie, Bing Wang, and Xiang Bai. Orion: A holistic end-to-end autonomous driving framework by vision-language instructed action generation. In *Proc. of IEEE Intl. Conf. on Computer Vision*, pp. 24823–24834, 2025.

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 *Proc. of the Conference on Robot Learning*, pp. 991–1002. PMLR, 2022.

Boseong Jeon, Yunho Choi, and Taehan Kim. Shallow-π: Knowledge distillation for flow-based vlas. In *Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems*, 2026.

Titong Jiang, Xuefeng Jiang, Yuan Ma, Xin Wen, Bailin Li, Kun Zhan, Peng Jia, Yahui Liu, Sheng Sun, and Xianpeng Lang. The better you learn, the smarter you prune: Towards efficient vision-language-action models via differentiable token pruning. *arXiv preprint arXiv:2509.12594*, 2025.

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. In *Proc. of the Conference on Robot Learning*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. In *Proc. of Robotics: Science and Systems*, 2025.

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.

Wei Li, Renshan Zhang, Rui Shao, Jie He, and Liqiang Nie. Cogvla: Cognition-aligned vision-language-action models via instruction-driven routing & sparsification. In *Proc. of Advances in Neural Information Processing Systems*, volume 38, pp. 137646–137675, 2025.

Haotian Liang, Xinyi Chen, Bin Wang, Mingkang Chen, Yitian Liu, Yuhao Zhang, Zanxin Chen, Tianshuo Yang, Yilun Chen, Jiangmiao Pang, et al. Mm-act: Learn from multimodal parallel generation to act. In *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pp. 35080–35090, 2026a.

Zhixuan Liang, Yizhuo Li, Tianshuo Yang, Chengyue Wu, Sitong Mao, Liuao Pei, Tian Nian, Shunbo Zhou, Xiaokang Yang, Jiangmiao Pang, et al. Discrete diffusion vla: Bringing discrete diffusion to action decoding in vision-language-action policies. In *Proc. of Intl. Conf. on Machine Learning*, 2026b.

Tao Lin, Yilei Zhong, Yuxin Du, Jingjing Zhang, Jiting Liu, Yinxinyu Chen, Encheng Gu, Ziyan Liu, Hongyi Cai, Yanwen Zou, et al. Evo-1: Lightweight vision-language-action model with preserved semantic alignment. In *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pp. 13397–13406, 2026.

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

Chaoqi Liu, Xiaoshen Han, Jiawei Gao, Yue Zhao, Haonan Chen, and Yilun Du. Oat: Ordered action tokenization. In *Proc. of Robotics: Science and Systems*, 2026.

Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *Proc. of European Conference on Computer Vision*, pp. 38–55. Springer, 2024.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In *Proc. of Intl. Conf. on Learning Representations*, 2025.

Yuxiang Lu, Zhe Liu, Xianzhe Fan, Zhenya Yang, Jinghua Hou, Junyi Li, Kaixin Ding, and Hengshuang Zhao. Faster: Rethinking real-time flow vlas. *arXiv preprint arXiv:2603.19199*, 2026.

Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. In *Proc. of Robotics: Science and Systems*, 2020.

Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*, 7(4):11205–11212, 2022a.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022b.

Jiahui Niu, Kefan Gu, Yucheng Zhao, Shengwen Liang, Tiancai Wang, Xing Hu, Ying Wang, and Huawei Li. Realtime-vla flash: Speculative inference framework for diffusion-based vlas. *arXiv preprint arXiv:2605.13778*, 2026.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, et al. Octo: An open-source generalist robot policy. In *Proc. of Robotics: Science and Systems*, 2024.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. In *Proc. of Robotics: Science and Systems*, 2025.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Proc. of the Conference on Robot Learning*, pp. 894–906. PMLR, 2022.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, et al. Smolvla: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

StarVLA Community. Starvla: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. In *Proc. of Advances in Neural Information Processing Systems*, volume 33, pp. 13139–13150, 2020.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, Guangzhong Sun, Xin Jin, and Zhibo Chen. Vla-jepa: Enhancing vision-language-action model with latent world model. In *Proc. of European Conference on Computer Vision*, 2026.

Hanzhen Wang, Jiaming Xu, Yushun Xiang, Jiayi Pan, Yongkang Zhou, Yong-Lu Li, and Guohao Dai. Specprune-vla: Accelerating vision-language-action models via action-aware self-speculative pruning. In *Proc. of Intl. Conf. on Machine Learning*, 2026a.

Qiuyue Wang, Mingsheng Li, Jian Guan, Jinhui Ye, Sicheng Xie, Yitao Liu, Junhao Chen, Zhixuan Liang, Jie Zhang, Xintong Hu, et al. Qwen-vla: Unifying vision-language-action modeling across tasks, environments, and robot embodiments. *arXiv preprint arXiv:2605.30280*, 2026b.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. In *Proc. of the AAAI Conf. on Artificial Intelligence*, pp. 18638–18646, 2026c.

Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. In *Proc. of Intl. Conf. on Learning Representations*, 2026d.

Junjie Wen, Yichen Zhu, Jinming Li, Minjie Zhu, Zhibin Tang, Kun Wu, Zhiyuan Xu, Ning Liu, Ran Cheng, Chaomin Shen, et al. Tinyvla: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025.

Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, Fangjing Wang, Qian Zhu, He Sun, Yong Wang, Shuailei Ma, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026a.

Xianjin Wu, Dingkang Liang, Tianrui Feng, Kui Xia, Yumeng Zhang, Xiaofan Li, Xiao Tan, and Xiang Bai. Generation models know space: Unleashing implicit 3d priors for scene understanding. In *Proc. of European Conference on Computer Vision*, 2026b.

Siyu Xu, Yunke Wang, Chenghao Xia, Dihao Zhu, Tao Huang, and Chang Xu. Vla-cache: Efficient vision-language-action manipulation via adaptive token caching. In *Proc. of Advances in Neural Information Processing Systems*, volume 38, pp. 164448–164473, 2025.

Yuhao Xu, Yantai Yang, Zhenyang Fan, Yufan Liu, Yuming Li, Bing Li, and Zhipeng Zhang. Qvla: Not all channels are equal in vision-language-action model's quantization. In *Proc. of Intl. Conf. on Learning Representations*, 2026.

Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, Junjin Xiao, Haoyun Liu, Ronghan Chen, Yuzhi Chen, Dongjie Huo, et al. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026a.

Yantai Yang, Yuhao Wang, Zichen Wen, Luo Zhongwei, Chang Zou, Zhipeng Zhang, Chuan Wen, and Linfeng Zhang. Efficientvla: Training-free acceleration and compression for vision-language-action models. In *Proc. of Advances in Neural Information Processing Systems*, volume 38, pp. 40891–40914, 2025.

Yi Yang, Xueqi Li, Yiyang Chen, Jin Song, Yihan Wang, Zipeng Xiao, Jiadi Su, You Qiaoben, Pengfei Liu, and Zhijie Deng. Mantis: A versatile vision-language-action model with disentangled visual foresight. In *Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition*, pp. 42505–42515, 2026b.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, et al. Gigaworld-policy: An efficient action-centered world–action model. *arXiv preprint arXiv:2603.17240*, 2026.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proc. of IEEE Intl. Conf. on Computer Vision*, pp. 11975–11986, 2023.

Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, He Wang, Zhizheng Zhang, et al. Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge. In *Proc. of Advances in Neural Information Processing Systems*, volume 38, pp. 24195–24228, 2025.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. In *Proc. of Robotics: Science and Systems*, 2023.

Jinliang Zheng, Jianxiong Li, Zhihao Wang, Dongxiu Liu, Xirui Kang, Yuchun Feng, Yinan Zheng, Jiayin Zou, Yilun Chen, Jia Zeng, et al. X-vla: Soft-prompted transformer as scalable cross-embodiment vision-language-action model. In *Proc. of Intl. Conf. on Learning Representations*, volume 2026, pp. 60580–60606, 2026.

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

A Implementation and Training Details

A.1 Architecture and Initialization

TurboVLA uses DINOv3 ViT-B/16 for LIBERO and ViT-L/16 for RoboTwin 2.0 (Siméoni et al., 2025), together with BERT-base-uncased (Devlin et al., 2019) for instruction encoding. The vision-language interaction module is initialized with pretrained feature-enhancement weights from the Grounding DINO Swin-T OGC checkpoint (Liu et al., 2024). Both configurations use a shared feature dimension of $d = 256$, six bidirectional interaction layers, and a three-layer ACT-style action decoder (Zhao et al., 2023). The current robot state is projected into $N_s = 2$ tokens, which are concatenated with the fused vision–language features to form the decoder memory. Table 7 summarizes the architecture configurations.

A.2 Data Processing

LIBERO. We jointly train on the four modified `no_noops` RLDS suites released with OpenVLA (Kim et al., 2024). Each sample contains a primary-view RGB image, a wrist-view RGB image, a task instruction, and an 8-dimensional robot state. We preserve the original 256×256 image resolution without resizing or cropping and apply the pixel normalization used by the pretrained visual encoder. Robot states are standardized using statistics computed over the combined training data. Each action is 7-dimensional: the six continuous arm-action coordinates are min–max normalized to $[-1, 1]$ and clipped to this range, while the gripper command retains its original binary encoding in $\{-1, +1\}$. Each target chunk contains $H = 12$ consecutive actions starting at the observation timestep. At episode boundaries, incomplete chunks are padded by repeating the final action, and padded entries are excluded from the loss.

RoboTwin 2.0. Training combines 2,500 clean and 25,000 randomized demonstrations across 50 tasks. We sample tasks uniformly, with randomized and clean demonstrations sampled at a ratio of $10 : 1$ within each task. Each sample contains a head-view RGB image, two wrist-view RGB images, a task instruction, and the current bimanual state. All three images have a resolution of 224×224. Both states and actions are 14-dimensional, comprising the left-arm joint positions, right-arm joint positions, and two gripper coordinates. Continuous joint coordinates are min–max normalized to $[-1, 1]$, while gripper coordinates are binarized using a threshold of 0.49. Each observation is paired with a target chunk of $H = 50$ consecutive absolute joint-position and gripper actions.

A.3 Training Configurations

We train both policies using AdamW to minimize an ℓ_1 behavior-cloning loss. BERT remains frozen, while the visual encoder and all remaining policy modules are optimized for action prediction. Training runs for 40,000 steps with an effective batch size of 128 on 4 NVIDIA RTX 4090 GPUs for LIBERO, and 150,000 steps with an effective batch size of 512 on 8 NVIDIA H100 GPUs for RoboTwin 2.0. Both benchmarks use linear learning-rate warm-up followed by a constant learning rate. We maintain an exponential moving average (EMA) of the trainable parameters with a decay factor of 0.999. Table 8 summarizes the training configurations.

B Evaluation Protocols and Efficiency Measurement

B.1 Evaluation Protocols

LIBERO. We evaluate the jointly trained policy on the Spatial, Object, Goal, and Long suites following the VLA-Adapter rollout protocol (Wang et al., 2026c). Each suite contains ten tasks, with 50 trials per task, yielding 2,000 evaluation trials in total. We use benchmark-provided initial states and a fixed evaluation seed, with ten settling steps at the start of each episode. Episode lengths are limited to 220, 280, 300, and 520 control steps for Spatial, Object, Goal, and Long, respectively, excluding the settling steps. The policy predicts 12-step action chunks and executes each chunk before replanning. We report the success rate for each suite and the average across the four suites.

RoboTwin 2.0. We evaluate the jointly trained policy on all 50 tasks under clean and randomized conditions using the StarVLA evaluation interface (StarVLA Community, 2026). We conduct 100

Table 7: Architecture configurations of TurboVLA.

Configuration	Value
Modality representations	
Shared feature dimension	256
Number of state tokens	2
vision-language interaction	
Number of layers	6
Attention heads	4
Internal dimension	1,024
Attention dropout	0
Stochastic-depth rate	0.1
Action decoder	
Number of layers	3
Attention heads	8
Feed-forward dimension	2,048
Output MLP hidden dimension	512
Dropout	0.1

Table 8: Training configurations of TurboVLA.

Configuration	LIBERO	RoboTwin 2.0
Optimizer	AdamW	AdamW
Peak learning rate	5×10^{-5}	5×10^{-5}
AdamW (β_1, β_2)	$(0.9, 0.95)$	$(0.9, 0.95)$
AdamW ϵ	10^{-8}	10^{-8}
Weight decay	10^{-10}	10^{-10}
Training steps	40,000	150,000
Warm-up steps	10,000	1,000
Effective batch size	128	512
Gradient clipping norm	1.0	1.0
EMA decay	0.999	0.999

trials per task under each condition, yielding 5,000 trials per setting, and report success rates for both settings and their average. Evaluation uses unseen task instructions.

Real-world evaluation. We evaluate TurboVLA on the AgileX Piper platform across four language-conditioned manipulation tasks, with 40 trials per task and 160 trials per policy. Before each trial, the robot is automatically reset to a fixed home configuration, and task objects are randomly positioned within a 50×50 cm region on the tabletop. Each trial has a time limit of 30 s, during which the policy may retry unsuccessful attempts. A trial ends upon task completion, expiration of the time limit, or an automatic safety stop triggered by a workspace-boundary violation. We compare against $\pi_{0.5}$ using the same robotic platform, fine-tuning data, and evaluation protocol, and report per-task success rates and their average. For *grab roller*, success requires grasping the target object and lifting it clear of the tabletop. For *move playing card away*, the robot must grasp and lift the card, then move it to the right. For *press stapler*, the robot must press the stapler down with its gripper closed. For *stack three bowls*, all three bowls must remain stably stacked after the gripper releases the final bowl. A trial is successful if the corresponding criterion is satisfied within the time limit.

B.2 Efficiency Measurement

We evaluate all models on a single NVIDIA RTX 4090 with batch size one, using official implementations in evaluation mode with gradients disabled. Within each benchmark, all models use the same action horizon: 12 for LIBERO and 50 for RoboTwin 2.0. Autoregressive policies generate the action tokens required for the specified horizon.

Latency. We report steady-state local policy inference latency, measured from GPU-resident observations to a complete action chunk on the GPU. This includes visual and language encoding, action generation, and internal action postprocessing, while excluding input preparation and transfer, communication, and environment execution. We report the median latency over 50 synchronized inference runs after ten warm-up iterations, with initial compilation excluded.

Algorithm 1 Latency and GPU memory measurement

Require: Model configuration c, observation specification ω, action horizon H
Ensure: Latency statistics and inference VRAM
Record idle GPU memory M_{idle} with `nvidia-smi`
Start periodic GPU memory sampling $\{M_t\}$
Launch benchmark process; initialize policy π and GPU-ready observation o from c, ω
Set π to evaluation mode with gradients disabled
Complete initial compilation, if required
for $i = 1, \ldots, 10$ **do**
    $A \leftarrow \pi(o; H)$
    Synchronize GPU execution
    Release A
end for
$\mathcal{T} \leftarrow [\,]$
for $i = 1, \ldots, 50$ **do**
    Synchronize GPU execution
    $t_0 \leftarrow$ TIME
    $A \leftarrow \pi(o; H)$ ▷ Complete action chunk on GPU
    Synchronize GPU execution
    Append TIME $- t_0$ to $\mathcal{T}$
    Release A
end for
Wait for the benchmark process to exit; stop memory sampling
$M_{\text{VRAM}} \leftarrow \max_t M_t - M_{\text{idle}}$
return $\text{median}(\mathcal{T})$, $\text{P90}(\mathcal{T})$, $\text{mean}(\mathcal{T})$, M_{VRAM}

GPU memory. Inference VRAM is the difference between the peak `nvidia-smi` reading during model initialization, warm-up, and timed inference and the reading on an idle GPU before the run. Memory usage is expressed in GiB (1024^3 bytes). Algorithm 1 summarizes the measurement procedure.

Parameter count. We count all parameters in the complete inference model, including both trainable and frozen parameters. Optimizer states are excluded.

C Additional Results and Ablations

C.1 Additional Ablation Studies

We report additional ablations on LIBERO and RoboTwin 2.0. LIBERO results examine interaction depth and action horizon; RoboTwin 2.0 results examine cross-modal interaction and the instruction encoder.

Interaction depth on LIBERO. We first examine how the number of vision-language interaction layers affects performance, keeping the remaining architecture fixed. As shown in Tab. 9, average success increases from 95.4% at $N = 2$ to 96.9% at $N = 4$ and 97.6% at $N = 6$. The improvement is most pronounced on LIBERO-Goal, where success rises from 90.4% to 98.2%, whereas the Object suite remains near saturation across all three depths. These results suggest that additional interaction layers are particularly useful for tasks requiring stronger instruction-conditioned visual grounding. We use $N = 6$, which achieves the highest average success among the evaluated depths.

Table 9: Depth ablation on LIBERO.

N	Params (M)	Spa.	Obj.	Goal	Long	Avg.
2	206.6	98.0	99.6	90.4	93.4	95.4
4	211.3	98.0	99.2	96.0	94.4	96.9
6	216.1	97.0	100.0	98.2	95.2	**97.6**

Action horizon on LIBERO. We next vary the predicted action horizon H while keeping the remaining architecture fixed. As shown in Fig. 6, average success increases from 96.8% at $H = 8$ to 97.3% at $H = 10$ and 97.6% at $H = 12$, before decreasing to 96.7% at $H = 15$. In the LIBERO rollout, the entire predicted chunk is executed before the policy receives a new observation. A longer horizon therefore extends the period of open-loop execution as well as the prediction window. The decline at $H = 15$ indicates that increasing the chunk length does not necessarily improve task performance under this execution protocol. We select $H = 12$ as the best of the evaluated settings.

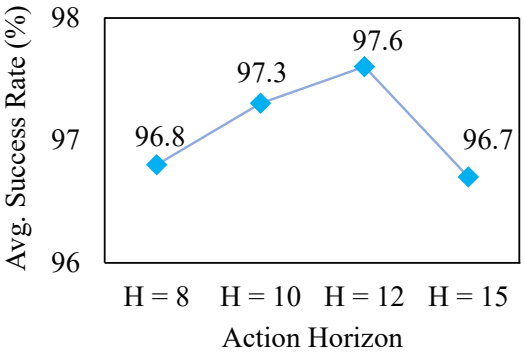


Figure 6: Action-horizon ablation on LIBERO.

Vision-language interaction on RoboTwin 2.0. We repeat the interaction-design comparison from Sec. 4.3 on RoboTwin 2.0 to test whether its effect extends to bimanual manipulation. As shown in Tab. 10, removing interaction yields 82.58% average success, compared with 88.06% for bidirectional interaction. Both one-way variants recover much of this gap: language queries visual features achieve 86.62%, while visual features querying language achieve 87.08%. Bidirectional interaction performs best under both clean and randomized conditions, reaching 88.02% and 88.10%, respectively. The consistent advantage over the one-way variants supports updating both feature streams before action prediction, as observed on LIBERO.

Instruction encoder on RoboTwin 2.0. Finally, we replace BERT with T5-Base or SigLIP-Base while retaining the same policy design. As shown in Tab. 11, T5-Base achieves 87.39% average success, compared with 88.06% for BERT, while SigLIP-Base reaches 86.05%. T5-Base matches BERT under the clean condition (88.04% versus 88.02%) but performs less well under randomization (86.74% versus 88.10%). Thus, the architecture remains effective with different instruction encoders, although encoder choice affects performance under the evaluated randomized conditions.

Table 10: Effect of vision-language interaction on RoboTwin 2.0.

Interaction Design	Clean	Rand.	Avg.
w/o Interaction	80.14	85.02	82.58
Language Queries Visual	86.28	86.96	86.62
Visual Queries Language	87.16	87.00	87.08
Bidirectional Interaction	88.02	88.10	**88.06**

Table 11: Effect of the instruction encoder on RoboTwin 2.0.

Text Encoder	Clean	Rand.	Avg.
SigLIP-Base	85.68	86.42	86.05
T5-Base	88.04	86.74	87.39
BERT	88.02	88.10	**88.06**

C.2 Per-Task Results on RoboTwin 2.0

Table 12 reports the success rates for TurboVLA on all 50 RoboTwin 2.0 tasks under clean and randomized conditions, with 100 evaluation trials per task in each condition. The average success rates are 88.02% and 88.10%, respectively.

C.3 Per-Task Results on LIBERO

Table 13 reports the success rates for TurboVLA on the 40 subtasks from the LIBERO Object, Goal, Spatial, and Long suites. Each suite contains ten tasks indexed by task ID from 0 to 9, with 50 evaluation trials per task. The final column reports the average success rate over the ten tasks in each suite.

TurboVLA achieves average success rates of 100.0%, 98.2%, 97.0%, and 95.2% on the Object, Goal, Spatial, and Long suites, respectively. Across all four suites, TurboVLA obtains 1,952 successes out of 2,000 trials, corresponding to an overall success rate of 97.6%.

Table 12: Per-task success rates (%) of TurboVLA on RoboTwin 2.0. Both conditions use the EMA checkpoint at 150k training steps with an action horizon of 50.

Task	Clean	Rand.	Task	Clean	Rand.
Adjust Bottle	100	99	Place Can Basket	72	77
Beat Block Hammer	100	94	Place Cans Plasticbox	100	99
Blocks Ranking RGB	89	88	Place Container Plate	98	97
Blocks Ranking Size	74	78	Place Dual Shoes	76	82
Click Alarmclock	64	73	Place Empty Cup	100	100
Click Bell	97	98	Place Fan	99	97
Dump Bin Bigbin	99	99	Place Mouse Pad	85	79
Grab Roller	99	100	Place Object Basket	86	78
Handover Block	86	78	Place Object Scale	93	94
Handover Mic	92	95	Place Object Stand	93	97
Hanging Mug	61	60	Place Phone Stand	77	95
Lift Pot	100	99	Place Shoe	100	100
Move Can Pot	93	89	Press Stapler	96	97
Move Pillbottle Pad	97	96	Put Bottles Dustbin	68	86
Move Playingcard Away	95	99	Put Object Cabinet	70	56
Move Stapler Pad	72	65	Rotate QR Code	94	95
Open Laptop	98	99	Scan Object	93	93
Open Microwave	80	90	Shake Bottle	99	99
Pick Diverse Bottles	81	83	Shake Bottle Horizontally	100	99
Pick Dual Bottles	87	89	Stack Blocks Three	87	76
Place A2B Left	79	73	Stack Blocks Two	98	98
Place A2B Right	72	58	Stack Bowls Three	89	87
Place Bread Basket	85	81	Stack Bowls Two	99	100
Place Bread Skillet	88	87	Stamp Seal	82	85
Place Burger Fries	98	100	Turn Switch	61	69
Average				**88.02**	**88.10**

Table 13: Per-task success rates (%) of TurboVLA on the four LIBERO suites.

Object	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Avg.**
TurboVLA	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Goal	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Avg.**
TurboVLA	100.0	100.0	100.0	94.0	100.0	96.0	98.0	100.0	100.0	94.0	98.2
Spatial	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Avg.**
TurboVLA	100.0	98.0	100.0	96.0	90.0	96.0	94.0	100.0	98.0	98.0	97.0
Long	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Avg.**
TurboVLA	86.0	100.0	94.0	98.0	98.0	92.0	92.0	100.0	94.0	98.0	95.2

C.4 Qualitative Rollouts

Figures 7–9 show selected rollouts of TurboVLA in simulation and on the real robot. The LIBERO examples span all four suites. The RoboTwin 2.0 examples cover cabinet manipulation, stapler placement, three-block stacking, and a two-arm handover. The real-world sequences provide a closer view of the four tasks summarized in Fig. 4.

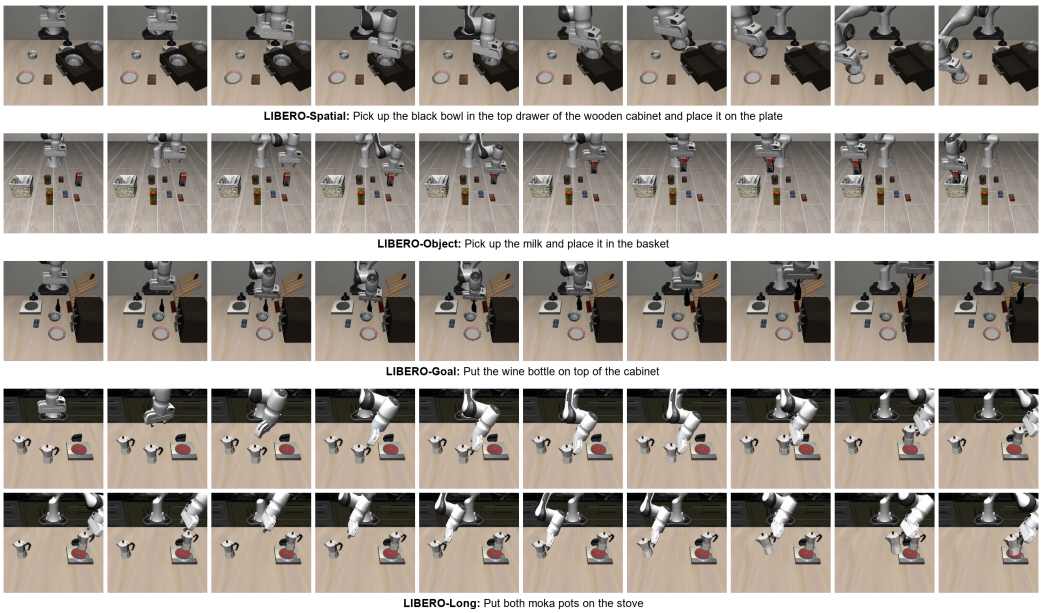


Figure 7: Qualitative rollouts on LIBERO. From top to bottom, the sequences show tasks from the Spatial, Object, Goal, and Long suites. Task instructions appear below their corresponding sequences.

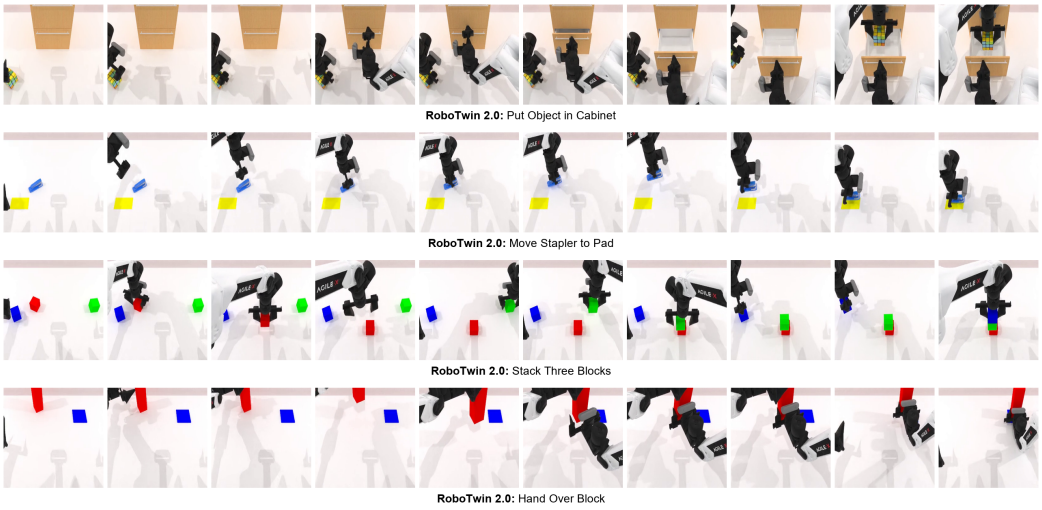


Figure 8: Qualitative rollouts on RoboTwin 2.0. From top to bottom, the sequences show putting an object in a cabinet, moving a stapler to a pad, stacking three blocks, and handing over a block.

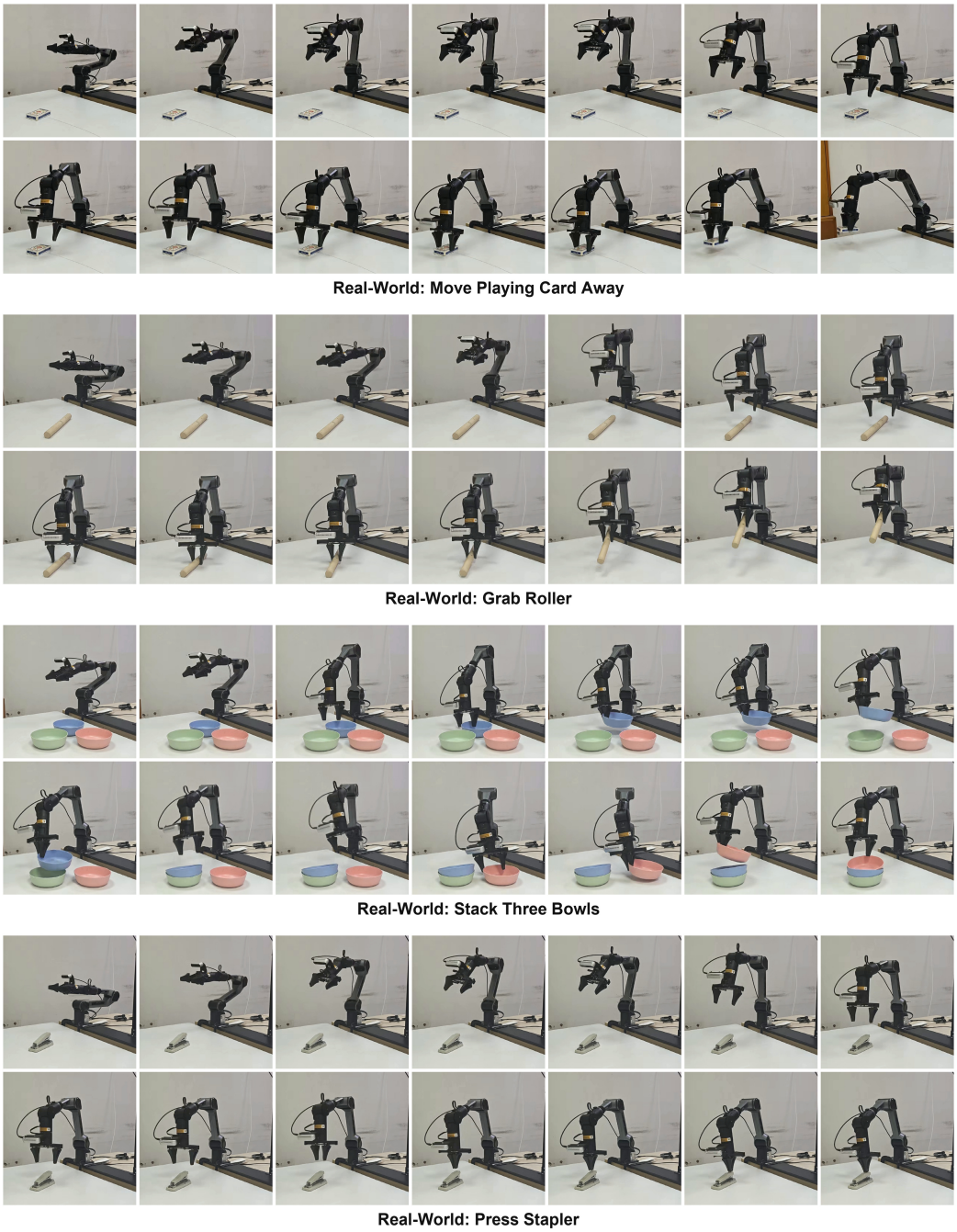


Figure 9: Real-world rollouts on the AgileX Piper platform. From top to bottom, the sequences show moving a playing card away, grasping a roller, stacking three bowls, and pressing a stapler.