VLA-Adapter: An Effective Paradigm for Tiny-Scale Vision-Language-Action Model

Yihao Wang[1,2,4,*,†], **Pengxiang Ding**[2,3,4,‡,†], **Lingxiao Li**[1,4,5,†], **Can Cui**[2,4], **Zirui Ge**[3,4], **Xinyang Tong**[2,4], **Wenxuan Song**[4,6], **Han Zhao**[2,3,4], **Wei Zhao**[2,4], **Pengxu Hou**[6], **Siteng Huang**[2], **Yifan Tang**[1], **Wenhui Wang**[1], **Ru Zhang**[1,§], **Jianyi Liu**[1], **Donglin Wang**[2,§]

[1]Beijing University of Posts and Telecommunications
[2]Westlake University
[3]Zhejiang University
[4]OpenHelix Team
[5]State Key Laboratory of Networking and Switching Technology
[6]The Hong Kong University of Science and Technology (Guangzhou)
yh-wang@bupt.edu.cn; dingpx2015@gmail.com

Abstract

Vision-Language-Action (VLA) models typically bridge the gap between perceptual and action spaces by pre-training a large-scale Vision-Language Model (VLM) on robotic data. While this approach greatly enhances performance, it also incurs significant training costs. In this paper, we investigate how to effectively bridge vision-language (VL) representations to action (A). We introduce VLA-Adapter, a novel paradigm designed to reduce the reliance of VLA models on large-scale VLMs and extensive pre-training. To this end, we first systematically analyze the effectiveness of various VL conditions and present key findings on which conditions are essential for bridging perception and action spaces. Based on these insights, we propose a lightweight Policy module with Bridge Attention, which autonomously injects the optimal condition into the action space. In this way, our method achieves high performance using only a 0.5B-parameter backbone, without any robotic data pre-training. Extensive experiments on both simulated and real-world robotic benchmarks show that VLA-Adapter not only achieves state-of-the-art level performance, but also offers the fast inference speed reported to date. Furthermore, thanks to the proposed advanced bridging paradigm, VLA-Adapter enables the training of a powerful VLA model on a single consumer-grade GPU, greatly lowering the barrier to deploying VLA model.

Project page — https://vla-adapter.github.io/
Appendix — https://arxiv.org/pdf/2509.09372

1 Introduction

In the past two years, with significant breakthroughs in multimodal LLMs (Karamcheti et al. 2024; Steiner et al. 2024; Li et al. 2025), developing robot systems with general perception, understanding, and behavior capabilities has become a key research direction in artificial intelligence. In particular, the emergence of the Vision-Language-Action (VLA) model offers a new solution for enabling robot operations driven by instructions (Kim et al. 2024; Zhao et al. 2025b; Kim, Finn, and Liang 2025; Song et al. 2025b; Cen et al. 2025; Zhang et al. 2025). Research on VLA primarily focuses on extracting multimodal information and aligning it with the action decision space to generate the high-quality actions (Team et al. 2024; Zhong et al. 2025).

Current VLA models typically require large-scale embodied data (e.g., DROID (Khazatsky et al. 2024)) to pre-train Multimodal Large Language Models (MLLMs) (Especially, VLMs) for task adaptability, which is then passed to the designed Policy network (Bu et al. 2024) to decode or generate actions for handling the tasks in the diverse environments (Liu et al. 2023; Mees et al. 2022).

However, VLA models still face several bottlenecks, including reliance on large-scale VLMs, slow fine-tuning speed, high GPU memory (VRAM) consumption, and low inference efficiency (throughput), as shown in Figure 1. To this end, it is necessary to explore the most essential but rarely discussed question in the VLA: ***How to bridge the***

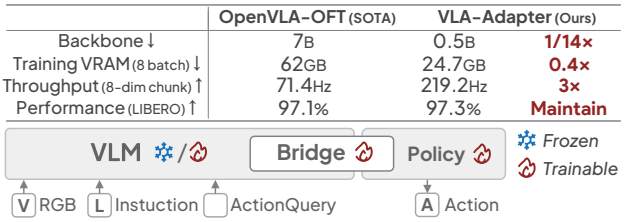


Figure 1: Characteristics of VLA-Adapter. "↓" is that smaller values are better, and vice versa. Our paradigm can quickly train the SOTA-level VLA using a tiny backbone.

[*]Work done during interning at Westlake University
[†]These authors contributed equally.
[‡]Project Lead
[§]Corresponding author

gap of VL (vision language) to A (action) more effectively?

To answer this question, we propose VLA-Adapter, a novel bridging paradigm for VLA. We systematically explore how different conditions influence action generation and give some key findings for VLA design. On this basis, we built a Policy with Bridge Attention to autonomously inject the optimal condition into the action space. Experiments show that VLA-Adapter has superior performance, high efficiency, and fast throughput. It lowers the barrier to VLA deployment. The main contributions are summarized.

- To our knowledge, this work is the first systematic analysis of bridging paradigms' effects on action generation. And we also give some key findings of the VLA design.
- VLA-Adapter transfers the sufficient multimodal information to the proposed Policy for action generation, effectively bridging the modality gap from VL to A.
- Experiments show that VLA-Adapter has a higher success rate, smaller scale, lower tuning cost, and faster inference in diverse simulated and real-world robotic tasks.

2 Related Work

2.1 Vision-Language-Action (VLA) Models

Recently, leveraging pre-trained Vision-Language Models (VLMs) (Karamcheti et al. 2024; Steiner et al. 2024; Li et al. 2025) to control robots for performing various daily tasks has substantially accelerated research in embodied intelligence. This has emerged as a prominent research focus (Intelligence 2025b; NVIDIA et al. 2025; Fan et al. 2025; Tong et al. 2025). These models are referred to as VLA.

Typically, VLA models require large-scale embodied datasets, such as Open X-Embodiment (Collaboration 2024), for pre-training (Liu et al. 2025). This process integrates VLMs with a dedicated Policy (Song et al. 2025a; Li et al. 2024b), allowing the system to decode or generate action sequences for diverse tasks in an end-to-end manner. Moreover, dual-system VLA (Shentu et al. 2024; Bu et al. 2024) have recently garnered attention. These methods introduce an intermediate latent token to connect the VLMs and Policy, using an asynchronous mechanism to enhance coordination between two systems (Zhang et al. 2024a).

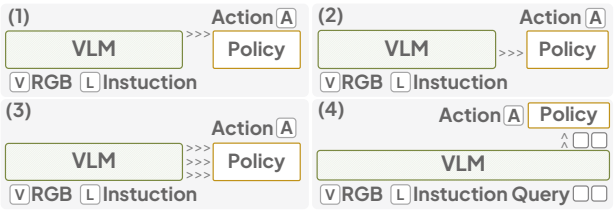
Consequently, effectively and efficiently bridging the vision-language perception space to the action space has become a key design challenge in the design of VLA models.

2.2 Bridging from Perception to Action Space

Earlier studies (Kim et al. 2024; Brohan et al. 2023a,b) attempted to directly align perception and action spaces by discretizing actions into tokens. However, this discretization inevitably introduces inherent loss. Recent studies have shifted their focus toward continuous action spaces (NVIDIA et al. 2025; Intelligence 2025a; Shukor et al. 2025). Based on the types of perceptual features utilized to bridge to the action space, they can be categorized:

1) Raw Features from VLMs. Raw features refer to vision and language representations, and they are extracted directly from the VLM. Early methods in this category extract representations from the final-layer VLM, operating under the assumption that it encodes the most task-relevant semantic information (Li et al. 2024a). More recent methods leverage the intermediate-layer features within the VLM (Intelligence 2025a). They believe that such representations may retain richer spatial and multimodal information, thereby benefiting Policy in tasks that demand fine-grained perception or complex reasoning. For example, some studies use features from a middle layer (NVIDIA et al. 2025), the first-half layers (Shukor et al. 2025), or all intermediate-layer features (Intelligence 2025a).

2) Additional Query as Interface. Furthermore, recent studies (Kim, Finn, and Liang 2025; Cui et al. 2025) have introduced a novel interface that employs additional queries as bridges between VLMs and Policy, rather than directly transmitting Raw features. These queries are learnable and can incorporate multimodal information, showing superior performance. The bridge paradigms are shown in Figure 2.



Representative work	Feature type	Which layer
(1) RoboVLMs	Raw features from VLMs	Last
(2) GR00T N1	Raw features from VLMs	Intermidiate
(3) π_0	Raw features from VLMs	All
(4) OpenVLA-OFT	Additional Query as Interface	Last

Figure 2: Existing bridge paradigms from VL to A.

3 VLA-Adapter Methodology

3.1 Preliminary

We present the VLA-Adapter framework, as illustrated in Figure 3. This VLM follows the Prismatic-VLMs architecture (Karamcheti et al. 2024). It has M layers. At timestep t, the input into VLM consists of $\{\mathcal{X}_t^v, \mathcal{X}_t^g, \mathcal{L}_t, \mathcal{AQ}_t\}$: the 3rd-view image $\mathcal{X}_t^v$, the wrist image $\mathcal{X}_t^g$, the instruction $\mathcal{L}_t$, and additional ActionQuery $\mathcal{AQ}_t$. After inputting $\mathcal{X}_t^v$ and $\mathcal{X}_t^g$, the DINOv2 (Oquab et al. 2024) and SigLIP (Zhai et al. 2023) extract vision embeddings. $\mathcal{L}_t$ is tokenized. The outputs are the specified-layer Raw latent $\mathcal{C}_t^{\mathcal{R}}$ and ActionQuery latent $\mathcal{C}_t^{\mathcal{AQ}}$. They serve as the conditions for Policy.

Backbone. To build a solid basis for research, we perform experiments of VLA-Adapter on different-scale backbones. The backbones select the Prismatic VLM trained on Qwen2.5-0.5B (Yang et al. 2024), the Prismatic VLM trained on LLaMA2-7B (Touvron et al. 2023), and OpenVLA-7B pre-trained on robotic data (Kim et al. 2024). The benefit gained from increasing backbone scale is limited in VLA-Adapter. The specific results are shown in Table 2 of **Section 4.1**. Therefore, to ensure efficiency, Qwen2.5-0.5B is our default backbone unless otherwise specified.

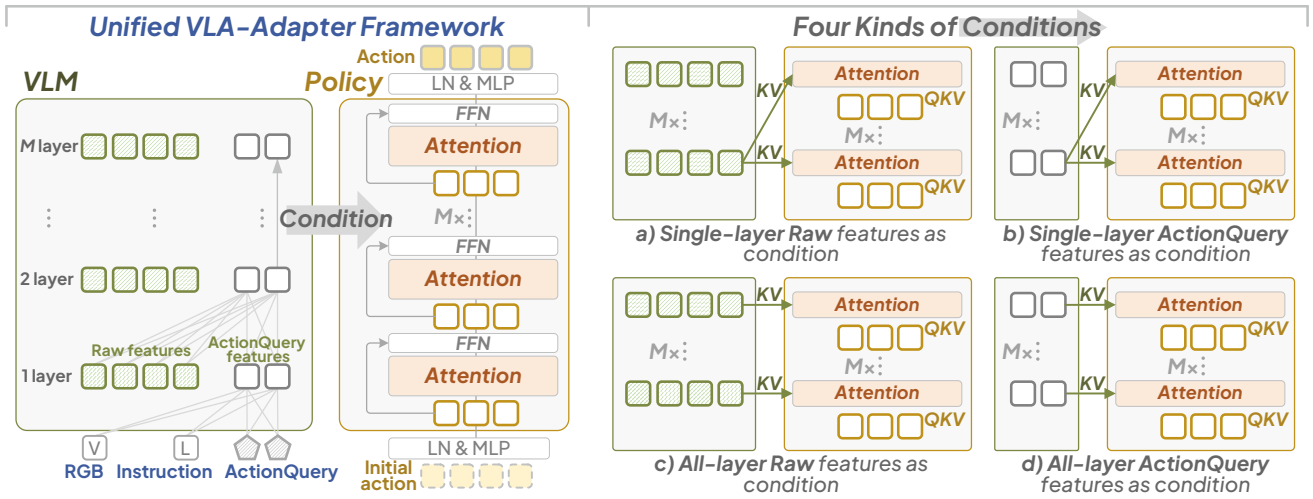


Figure 3: The proposed VLA architecture. The key components are the effective condition exploration and Attention design. "*Attention*" specifically includes cross attention with conditions and self attention with itself. In the VLA-Adapter framework, "*Attention*" is the Bridge Attention as shown in **Section 3.3**. Four conditions about "layer" and "type" are given on the right.

3.2 Which Condition Is Essential for Bridging from Perception to Action Space?

The relative effectiveness of the existing methods remains inconclusive. This is mainly due to the differences in the design of the VLM backbone and the Policy network. To address this gap, we explore which type of perception information is essential for action generation in the Policy network. In summary, we mainly focus on the following questions:

Question 1.1. *Which layer of features within the VLM is more effective for the Policy network?*

Question 1.2. *Are the ActionQuery features a better choice than the raw (vision-language) features?*

To ensure compatibility with existing experimental protocols for representative work (e.g., π_0), we let the number of Policy layers be equal to the number of VLM layers. At each layer of Policy, the action latent undergoes cross-attention with conditions and self-attention with itself. Details of the Policy with Attention can be found in the **Section 3.3**.

Experimental Setting. We evaluate four conditions in our framework. For *Question 1.1*, to evaluate the effectiveness of the individual-layer information, we employ the single-layer latent as the conditions for the all-layer Policy, as shown in Figure 3a) and 3c). To evaluate the effectiveness of all-layer information, we feed each-layer latent into the corresponding-layer Policy, as shown in Figure 3b) and 3d). For *Question 1.2*, to compare the effectiveness of the feature types, we use the $\mathcal{C}_t^{\mathcal{R}}$ or $\mathcal{C}_t^{\mathcal{AQ}}$ as conditions. The comparison on the LIBERO-Long (Liu et al. 2023), which is the long-horizon and complex benchmark, the results are as shown in Figure 4. And then, we give the following key findings.

Key Finding 1.1. *Regarding Raw features, the middle-layer latent performs better than the deep-layer latent.* Deep-layer $\mathcal{C}_t^{\mathcal{R}}$ is biased towards semantic information and less effective in action generation.

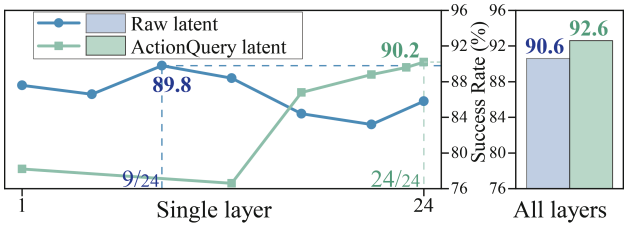


Figure 4: Comparison of four conditions in VLA-Adapter. The results are shown in **Appendix C**. The number of ActionQuery is 64. It is not fixed, we explore it in **Section 4.5**.

Key Finding 1.2. *Regarding ActionQuery features, deep-layer latent performs better than other-layer latent.* ActionQuery is trained from scratch, and deep-layer $\mathcal{C}_t^{\mathcal{AQ}}$ aggregates richer multimodal details.

Key Finding 1.3. *Multi-layer features perform better.* We observed that using all-layer features outperforms a single layer. Not only does it improve performance, but it also saves time on best layer selection during design.

$\mathcal{C}_t^{\mathcal{R}}$	9	13	$\mathcal{C}_t^{\mathcal{AQ}}$	1	13	17	21	23	24	All
Subtask 7	**90**	82	Subtask 7	76	66	74	70	70	74	76
Subtask 9	74	**84**	Subtask 9	78	62	58	72	72	84	78

Table 1: Comparison of the ith-layer $\mathcal{C}_t^{\mathcal{R}}$ and $\mathcal{C}_t^{\mathcal{AQ}}$ in subtasks of LIBERO-Long. The value is Success Rate (%).

Condition Determination. Does VLA-Adapter rely exclusively $\mathcal{C}_t^{\mathcal{AQ}}$ as the condition? The answer is no. While all-layer $\mathcal{C}_t^{\mathcal{AQ}}$ outperforms $\mathcal{C}_t^{\mathcal{R}}$, middle-layer $\mathcal{C}_t^{\mathcal{R}}$ excels in some hard tasks. Comparison is shown in Table 1. So, we aim to enhance performance by using certain knowledge from $\mathcal{C}_t^{\mathcal{R}}$.

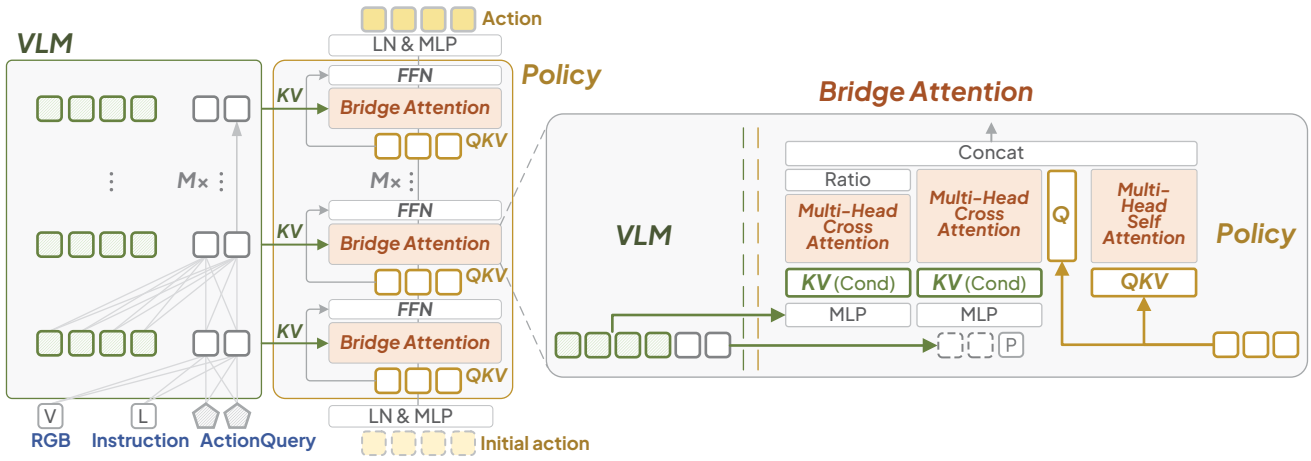


Figure 5: The Policy with Bridge Attention. The Policy parameters are only 97M when the backbone is Qwen2.5-0.5B. Each-layer $\mathcal{C}_t^{\mathcal{R}}$ and $\mathcal{C}_t^{\mathcal{AQ}}$ are integrated in Bridge Attention with the corresponding-layer action latent. Bridge Attention maps VL to Action to the greatest extent. The degree of $\mathcal{C}_t^{\mathcal{R}}$ injection is learnable, ensuring the performance and stability of training.

3.3 Policy with Bridge Attention

Overall. For the simplicity of the model, we designed an L1-based Policy network. At t-th timestep, the input to Policy includes: $\{\mathcal{C}_t^{\mathcal{R}}, \mathcal{C}_t^{\mathcal{AQ}}, \mathbf{A}_t^{\tau=0}, \mathcal{P}_t\}$. τ is the layer of Policy, and it has $\tau \in \mathbb{Z}^+$, $0 \leq \tau \leq M-1$. $\mathbf{A}_t^0$ is the H-step initial action of all zeros, it is processed by LayerNorm (LN) and Multi Layer Perceptron (MLP) to obtain $\widetilde{\mathbf{A}}_t^0 = [\widetilde{\mathbf{a}}_t^0, \widetilde{\mathbf{a}}_{t+1}^0, \ldots, \widetilde{\mathbf{a}}_{t+H-1}^0]$. $\mathcal{P}_t$ is the proprioceptive state, and it is mapped through a two-layer MLP to obtain the proprio embedding $\sigma_0(\mathcal{P}_t)$. The output is the H-step action chunk $\mathbf{A}_t^{M-1}$. Each layer is composed of a Bridge Attention module and a Feed-Forward Network (FFN). The Bridge Attention architecture is shown in Figure 5.

Bridge Attention. The Bridge Attention hopes to guide action generation to the greatest extent possible through the conditions $\mathcal{C}_t^{\mathcal{R}}$ and $\mathcal{C}_t^{\mathcal{AQ}}$. Each Bridge Attention consists of two cross attentions and one self attention. In the first cross attention, $\mathcal{C}_t^{\mathcal{R}}$ is processed through an MLP σ_1 to obtain K_1, V_1. The action latent $\widetilde{\mathbf{A}}_t^\tau$ is used as the Q_1, and perform attention to get $\text{CA}_1\left(\widetilde{\mathbf{A}}_t^\tau, \sigma_1(\mathcal{C}_t^{\mathcal{R}})\right)$. In the second cross attention, $\mathcal{C}_t^{\mathcal{AQ}}$ needs to be concatenated with the $\sigma_0(\mathcal{P}_t)$ and passed through an MLP σ_2 to obtain K_2, V_2. $\widetilde{\mathbf{A}}_t^\tau$ is used as the Q_2 to get $\text{CA}_2\left(\widetilde{\mathbf{A}}_t^\tau, \sigma_2\left[\mathcal{C}_t^{\mathcal{AQ}}, \sigma_0(\mathcal{P}_t)\right]\right)$. In the self attention, $\widetilde{\mathbf{A}}_t^\tau$ is as Q, K, V, and there is $\text{SA}(\widetilde{\mathbf{A}}_t^\tau, \widetilde{\mathbf{A}}_t^\tau)$.

To selectively inject certain $\mathcal{C}_t^{\mathcal{R}}$ into the action space of the Policy, we introduce a learning parameter Ratio g to modulate the influence of $\text{CA}_1\left(\widetilde{\mathbf{A}}_t^\tau, \sigma_1(\mathcal{C}_t^{\mathcal{R}})\right)$. g is initialized to 0 value, and the $\tanh$ activation function is utilized $\tanh(g) \in [-1, 1]$ to prevent extreme values from destabilizing the distribution (Zhang et al. 2024b). And then, the three attentions are concatenated to obtain $\widehat{\mathbf{A}}_t^\tau$.

After Bridge Attention, $\widehat{\mathbf{A}}_t^\tau$ passes through a residual FFN to obtain $\widetilde{\mathbf{A}}_t^{\tau+1}$. Repeating the above process, we obtain $\widetilde{\mathbf{A}}_t^{M-1}$. The action chunk $\mathbf{A}_t^{M-1}$ is get by an LN and MLP.

Additionally, we also design a DiT-based (Diffusion Transformer (Peebles and Xie 2023)) Policy. Since the diversity of Policy is not the focus of this paper, we put its details and the brief results in **Appendix B**. The results show that L1-based performance and inference speed are generally superior to those of the DiT-based approach. Therefore, VLA-Adapter chose the L1 architecture as the Policy.

3.4 Training

The training is conducted end-to-end, with the Policy trained from scratch. Given a ground truth action trajectory $\mathbf{A}_t$ and action latent $\mathbf{A}_t^\tau$. We train model $\pi_\theta(\cdot)$ with the objective:

$$\min_\theta \mathcal{J}(\theta) = \mathbb{E}_{\mathbf{A}_t, \mathcal{C}_t^{\mathcal{R}}, \mathcal{C}_t^{\mathcal{AQ}}, \sigma_0(\mathcal{P}_t)}\Big[\big\|\pi_\theta(\mathbf{A}_t^\tau, \mathcal{C}_t^{\mathcal{R}}, \mathcal{C}_t^{\mathcal{AQ}}, \sigma_0(\mathcal{P}_t)) - \mathbf{A}_t\big\|_1\Big]. \tag{1}$$

For more details of training, please see **Appendix F.1**.

4 Experiments

All experiments are run on a server equipped with 4 NVIDIA H100 GPUs. For more details of hyperparameters of VLA-Adapter, please see **Appendix F.2**. We perform experiments to answer the three questions:

Question 2.1. *What are the advantages of the VLA-Adapter compared to other bridge paradigms?* (Section 4.1)

Question 2.2. *How does VLA-Adapter perform compared to existing methods?* (Section 4.2, 4.3, and 4.4)

Question 2.3. *What else key components in the VLA-Adapter paradigm are worth exploring?* (Section 4.5)

Experiment Overview. In **Section 4.1**, we use the complex LIBERO-Long (Liu et al. 2023), which typically has a low success rate, to investigate the necessity of VLA-Adapter. From **Section 4.2** to **Section 4.4**, we use LIBERO

(Liu et al. 2023) and CALVIN (Mees et al. 2022), which are widely used in VLA, as well as real-world data, to comprehensively compare the overall performance. In **Section 4.5**, we use LIBERO-Long to explore key parts of VLA-Adapter.

4.1 Necessity of VLA-Adapter

Effectiveness. To validate the effectiveness of our bridge paradigm, we compare three kinds of backbones:

B1: The Prismatic VLM trained on Qwen2.5-0.5B.

B2: The Prismatic VLM trained on LLaMA2-7B.

B3: The OpenVLA-7B pre-trained on robotic data.

The first two are different-scale backbones without pre-training on robotic data. We adopted the OpenVLA-OFT bridging way (Kim, Finn, and Liang 2025) to compare. It is the SOTA on major benchmarks, including LIBERO-Long (Liu et al. 2023). The comparison is shown in Table 2.

Fine-tuned	*B1*		*B2*		*B3*	
	+OFT	+Ours	+OFT	+Ours	+OFT	+Ours
Success Rate ↑	85.8	**95.0**	87.5	**95.2**	94.5	**95.4**
Δ	9.2% ↑		7.7% ↑		0.9% ↑	

Table 2: Effectiveness comparison with OpenVLA-OFT on the LIBERO-Long. "Fine-tuned" is by LoRA tuning (Hu et al. 2022). **Bold** represents the best performance. Δ is the increment. Please note, comparison with bridge ways of π_0 (Intelligence 2025a) and GR00T N1 (NVIDIA et al. 2025) has been included in **Section 3**, so we will not elaborate here.

Fortunately, VLA-Adapter remains effective when the backbone is frozen. Only the ActionQuery latent and Policy are trained from scratch. SmolVLA (Shukor et al. 2025) is the VLA dedicated to studying frozen VLMs. So, we compare with OpenVLA-OFT and SmolVLA. The results are shown in Table 2. Since the results of GR00T N1 (NVIDIA et al. 2025) come from Hume (Song et al. 2025a), it did a full-params fine-tuning, so we will not compare with it here.

Frozen	OpenVLA-OFT	SmolVLA	**VLA-Adapter**
Success Rate (%) ↑	0.0	77.0	**86.4**

Table 3: Effectiveness comparison when the backbone is frozen. Benchmark is the same as Table 2. For an analysis of OpenVLA-OFT does not work, please see **Appendix H**.

Based on Tables 2 and 3, we summarize two conclusions:

Conclusion 1. *VLA-Adapter improvement effect is obvious when VLMs are not pre-trained on robotic data.*

Conclusion 2. *Even if the backbone freezes, VLA-Adapter still performs strongly, surpassing SmolVLA by 9.4%.*

This can be attributed to the fact that, after pre-training on robotic data, the last-layer features are already adapted to the action domain, enabling efficient fine-tuning with a simple MLP. However, when VLMs without pre-training, relying solely on the last-layer latents, are insufficient for effective action mapping. So, adopting the VLA-Adapter becomes crucial to achieve efficient fine-tuning. These insights highlight a ***key advantage***: VLA-Adapter facilitates efficient fine-tuning of VLMs without robotic pre-training, achieving performance that surpasses baselines using a tiny backbone.

Efficiency. VLA-Adapter attains a faster inference speed. The comparison is shown in Table 4.

Efficiency	OpenVLA-OFT	VLA-Adapter
Throughput (Hz) ↑	71.4	**219.2**
Latency (Sec) ↓	0.1120	**0.0365**

Table 4: Inference efficiency comparison. The action chunk is 8 steps, consistent with most VLA works.

4.2 Overall Performance on Various Tasks

Benchmark. We selected the widely adopted LIBERO benchmark (Liu et al. 2023) to evaluate performance across various types of tasks. LIBERO provides multiple suites, including Spatial, Object, Goal, and Long. For detailed settings and examples of LIBERO, please see **Appendix A**.

Baselines. We selected recently published, comprehensive SOTA VLA methods as baselines. They are large-scale: UnifiedVLA, OpenVLA, OpenVLA-OFT, UniVLA, CoT-VLA, and WorldVLA; Small-scale: SpatialVLA, π_0, π_0-FAST, SmolVLA, and GR00T N1; Tiny-scale: Seer, VLA-OS, and Diffusion Policy. Their results are all derived from original references or the reproduction of other published works, ensuring objectivity and accuracy.

Metrics. Each subtask is repeated multiple times (50 repetitions in this paper) to evaluate the model. We use the commonly used metric "Success Rate", reported as ranging from 0 to 100, with higher values indicating better performance.

Results. Comparison on the LIBERO is shown in Table 5. The results in Table 5 show that VLA-Adapter, using only a tiny-scale backbone, can achieve performance comparable to OpenVLA-OFT with 14× larger. It surpasses representative works such as π_0, SmolVLA, and GR00T N1. In addition, VLA-Adapter has a notable advantage of 29.0% over VLA-OS with the same-scale backbone on LIBERO-Long. These show the VLA-Adapter superiority on various tasks.

4.3 Performance on Generalization Tasks

Benchmark. We used the CALVIN ABC→D (Mees et al. 2022) to evaluate the performance on the zero-shot generalization tasks. CALVIN consists of four environments (Env A, B, C, and D) . "ABC→D" means it trains on Env A, B, and C and evaluates on Env D. VLA needs to execute a preset sequence of 1,000 tasks in sequence. Each task row consists of five subtasks. The model can only enter the processing of the next subtask after completing the current subtask. Please see **Appendix E** for more settings and examples.

Baselines. We selected methods recently as baselines. They are large-scale: UniVLA, OpenVLA, RoboDual, and OpenHelix; Small-scale: DeeR and VPP; Tiny-scale: Seer and MoDE. The results are based on original references or other published works, ensuring objectivity and correctness.

	LIBERO	Params	Spatial	Object	Goal	Long	Avg.
Large	UnifiedVLA (Wang et al. 2025) *(ArXiv)*	8.5	95.4	*98.8*	93.6	94.0	95.5
	OpenVLA (Kim et al. 2024) *(CoRL)*	7	84.7	88.4	79.2	53.7	76.5
	OpenVLA-OFT (Kim, Finn, and Liang 2025) *(RSS)*	7	*97.6*	98.4	**97.9**	*94.5*	*97.1*
	UniVLA (Bu et al. 2025) *(RSS)*	7	96.5	96.8	95.6	92.0	95.2
	CoT-VLA (Zhao et al. 2025a) *(CVPR)*	7	87.5	91.6	87.6	69.0	81.1
	WorldVLA (Cen et al. 2025) *(ArXiv)*	7	87.6	96.2	83.4	60.0	81.8
Small	SpatialVLA (Qu et al. 2025) *(RSS)*	4	88.2	89.9	78.6	55.5	78.1
	π_0 (Intelligence 2025a) *(RSS)*	3	96.8	*98.8*	95.8	85.2	94.2
	π_0-FAST (Pertsch et al. 2025) *(RSS)*	3	96.4	96.8	88.6	60.2	85.5
	SmolVLA (Shukor et al. 2025) *(ArXiv)*	2.2	93.0	94.0	91.0	77.0	88.8
	GR00T N1 (NVIDIA et al. 2025) *(ArXiv)*	2	94.4	97.6	93.0	90.6	93.9
Tiny	Seer† (Tian et al. 2025) *(ICLR)*	0.57	-	-	-	78.7	78.7
	VLA-OS (Gao et al. 2025) *(ArXiv)*	0.5	87.0	96.5	92.7	66.0	85.6
	Diffusion Policy† (Chi et al. 2023) *(RSS)*	-	78.3	92.5	68.3	50.5	72.4
	VLA-Adapter (Ours)	**0.5**	**97.8**	**99.2**	*97.2*	**95.0**	**97.3**
	VLA-Adapter-Pro (Ours)	**0.5**	**99.6***	**99.6***	**98.2***	**96.4***	**98.5***

Table 5: Comparison on the LIBERO. **Bold*** is the best performance, **Bold** is the suboptimal performance, and *Italics* is the third best performance. † represents that the non-based-VLM baselines. "Params" is the backbone scale, and its unit is "Billion". We give the performance on subtasks. It is shown in Table D1 of **Appendix D**. Recently, we have updated the "VLA-Adapter-Pro". Its Policy architecture is the same as Figure 5, and we optimized the implementation. For its details, please see **Appendix I**.

	CALVIN ABC→D	Params	Task completed in a row ↑ 1	2	3	4	5	Avg. len ↑
Large	UniVLA (Bu et al. 2025) *(RSS)*	7	95.5	85.8	75.4	66.9	56.5	3.80
	OpenVLA (Kim et al. 2024) *(CoRL)*	7	91.3	77.8	62.0	52.1	43.5	3.27
	OpenVLA-OFT (Kim, Finn, and Liang 2025) *(RSS)*	7	96.3	89.1	82.4	75.8	66.5	4.10
	RoboDual (Bu et al. 2024) *(ArXiv)*	7	94.4	82.7	72.1	62.4	54.4	3.66
	OpenHelix (Cui et al. 2025) *(ArXiv)*	7	*97.1**	91.4	82.8	72.6	64.1	4.08
Small	DeeR (Yue et al. 2024) *(NeurIPS)*	3	86.2	70.1	51.8	41.5	30.4	2.82
	VPP† (Hu et al. 2025) *(ICML)*	1.5	95.7	91.2	*86.3**	*81.0**	*75.0**	*4.33**
Tiny	Seer_{Large}† (Tian et al. 2025) *(ICLR)*	0.57	96.3	*91.6**	86.1	80.3	74.0	4.28
	MoDE† (Reuss et al. 2025) *(ICLR)*	0.44	96.2	88.9	81.1	71.8	63.5	4.01
	VLA-Adapter (Ours)	**0.5**	**99.1***	**94.6**	**88.8**	**82.8**	**76.5**	**4.42**
	VLA-Adapter-Pro (Ours)	**0.5**	**98.5**	**95.0***	**90.5***	**85.3***	**80.0***	**4.50***

Table 6: Comparison on the CALVIN ABC→D. **Bold*** is the best performance, **Bold** is the suboptimal performance, and *Italics* is the third best performance. † represents that the non-based-VLM method. Recently, we have updated "VLA-Adapter-Pro". Its Policy architecture is the same as Figure 5, and we optimized the implementation. For its details, please see **Appendix I**.

Metrics. We use the widely used "Success Rate" (the same in LIBERO) and "Avg. length" of completed tasks (the larger the better, with values between 0-5) as metrics.

Results. Comparison on the CALVIN is shown in Table 6. The results show that VLA-Adapter has strong generalization, and its Avg. length is better than SOTA baselines.

4.4 Performance on Real-World Tasks

Experimental Settings. We use a robotic system to perform real-world tasks. A 6-DOF Synria Alicia-D equipped with a 1-DOF wrist is employed, and it uses Logitech C920e and RealSense D405 cameras to capture 3rd-view and wrist images. The system is shown in Figure 6. We evaluate VLA-Adapter across four experimental categories:

1) *Simple pick-and-place tasks with objects spanning diverse materials and geometries.*

2) *CALVIN-inspired challenging task II: lateral block relocation (e.g. "Move <obj> left/right").*

3) *CALVIN-inspired challenging manipulation task I: "Block stacking".*

4) *LIBERO-inspired complex and long-horizon task: (e.g. "Pick up the spoon and place it on the cup, then place the cup on the plate").*

To strengthen evaluation rigor and assess generalization performance, we randomize the object positions at test time to induce distribution shift and increase task difficulty.

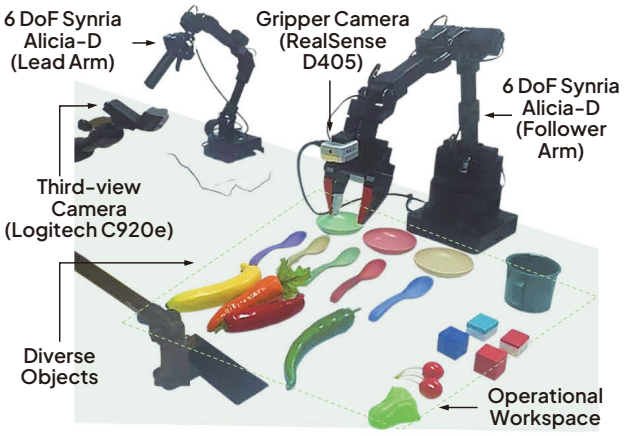


Figure 6: Real-world robotic system Synria Alicia-D.

Baselines. ACT (Zhao et al. 2023) and OFT-style variant (Kim, Finn, and Liang 2025) are as baselines.

Results. The comparison results are shown in Figure 7. Each result is obtained by averaging the results of 10 executions. Experimental results show that VLA-Adapter has better generalization capabilities in various scenarios. More real-world experiments are detailed in **Appendix G**.

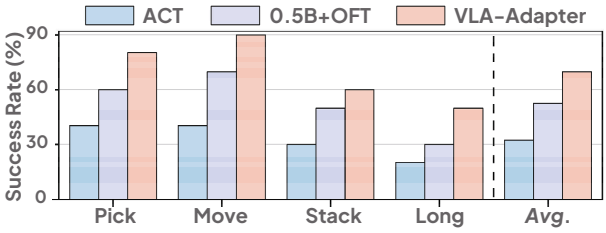


Figure 7: Comparison on real-world tasks.

4.5 Ablation Experiments

We explore three key components in the VLA-Adapter: 1. Number of ActionQuery, 2. Condition type, and 3. Injection degree for Policy. The benchmark is LIBERO-Long.

Number of ActionQuery. In our paradigm, the number of ActionQuery is not fixed. To explore the impact of this number on performance, we conducted the following experiments by varying the number of ActionQuery to 1, 4, 8, 16, 64, 128, 256, and 512. The results are shown in Figure 8. Thus, using too few ActionQuery tokens weakens multimodal aggregation and makes it challenging to condition the Policy. Conversely, employing too many ActionQuery tokens introduces redundancy, interfering with the performance. Therefore, we selected 64 ActionQuery tokens.

Condition Type. In **Section 3**, we analyzed the overall effects of different conditions on action generation. Here, we present the complete comparison results based on the four classic paradigms in **Section 2**, as shown in Table 7.

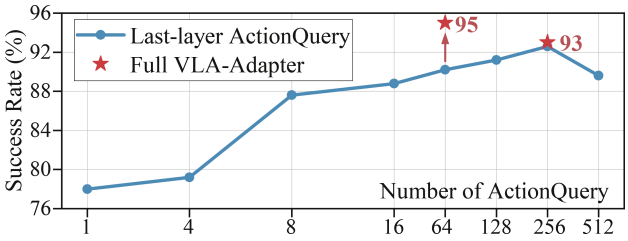


Figure 8: Comparison of the different numbers of ActionQuery. Blue line shows the result of using only ActionQuery. Red star shows the result of the full VLA-Adapter.

Layer	Raw	ActionQuery	Style	SR ↑
Last	✓	✗	RoboVLMs	85.8
	✗	✓	OpenVLA-OFT	90.2
Intermidiate	✓	✗	GR00T N1	88.4
All	✓	✗	π_0	90.6
	✗	✓	*N/A*	92.6
	✓	✓	**VLA-Adapter**	**95.0**

Table 7: Comparison with different condition types. The style can be summarized as representative works in Figure 2 of **Section 2**. "*N/A*" represents no such method.

Injection Degree for Policy. In the Bridge Attention, we use learnable parameters to control the injection degree of Raw features $\mathcal{C}_t^{\mathcal{R}}$ and set the injection degree of ActionQuery features $\mathcal{C}_t^{\mathcal{AQ}}$ to 1. Here, we explore other injection degrees, and the comparison results are shown in Table 8.

	Raw	ActionQuery	Success Rate (%)
1) **(VLA-Adapter)**	$\tanh(g)$	1	**95.0**
2)	1	1	91.4
3)	1	$\tanh(g)$	91.0
4)	$\tanh(g)$	$\tanh(g)$	92.6

Table 8: Ablation of other injection degrees.

5 Conclusion

We propose VLA-Adapter, a novel and efficient bridging paradigm for VLA. By leveraging Raw and ActionQuery latent, this method effectively transfers multimodal knowledge to the Policy to generate action. Experiments show that VLA-Adapter achieves SOTA performance using a tiny-scale backbone. In addition, our method has low VRAM usage and high inference speed. These results suggest that VLA-Adapter lowers the barrier to deploying VLA.

Ultimately, we hope the method and findings of this study can provide a solid basis for future research in the VLA and inspire the development of more advanced VLA methods!

Acknowledgments

This work was supported in part by the National Natural Science Foundation of China under Grant U21B2020.

References

Brohan, A.; Brown, N.; Carbajal, J.; Chebotar, Y.; Chen, X.; Choromanski, K.; Ding, T.; Driess, D.; Dubey, A.; Finn, C.; Florence, P.; Fu, C.; Arenas, M. G.; Gopalakrishnan, K.; Han, K.; Hausman, K.; Herzog, A.; and Hsu, J. 2023a. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *ArXiv*.

Brohan, A.; Brown, N.; Carbajal, J.; Chebotar, Y.; Dabis, J.; Finn, C.; Gopalakrishnan, K.; Hausman, K.; Herzog, A.; Hsu, J.; Ibarz, J.; Ichter, B.; Irpan, A.; Jackson, T.; Jesmonth, S.; Joshi, N. J.; and Julian, R. 2023b. RT-1: Robotics Transformer for Real-World Control at Scale. In *RSS*.

Bu, Q.; Li, H.; Chen, L.; Cai, J.; Zeng, J.; Cui, H.; Yao, M.; and Qiao, Y. 2024. Towards Synergistic, Generalized and Efficient Dual-System for Robotic Manipulation. In *ArXiv*.

Bu, Q.; Yang, Y.; Cai, J.; Gao, S.; Ren, G.; Yao, M.; Luo, P.; and Li, H. 2025. UniVLA: Learning to Act Anywhere with Task-Centric Latent Actions. In *RSS*.

Cen, J.; Yu, C.; Yuan, H.; Jiang, Y.; Huang, S.; Guo, J.; Li, X.; Song, Y.; Luo, H.; Wang, F.; Zhao, D.; and Chen, H. 2025. WorldVLA: Towards Autoregressive Action World Model. In *ArXiv*.

Chi, C.; Xu, Z.; Feng, S.; Cousineau, E.; Du, Y.; Burchfiel, B.; Tedrake, R.; and Song, S. 2023. Diffusion policy: Visuomotor Policy Learning via Action Diffusion. In *RSS*.

Collaboration, O. X.-E. 2024. Open X-Embodiment: Robotic learning datasets and RT-X models. In *ICRA*.

Cui, C.; Ding, P.; Song, W.; Bai, S.; Tong, X.; Ge, Z.; Suo, R.; Zhou, W.; Liu, Y.; Jia, B.; Zhao, H.; Huang, S.; and Wang, D. 2025. OpenHelix: A Short Survey and Empirical Analysis and Open-Source Dual-System VLA Model for Robotic Manipulation. In *ArXiv*.

Fan, Y.; Ding, P.; Bai, S.; Tong, X.; Zhu, Y.; Lu, H.; Dai, F.; Zhao, W.; Liu, Y.; Siteng Huang, Z. F.; Chen, B.; and Wang, D. 2025. Long-VLA: Unleashing Long-Horizon Capability of Vision Language Action Model for Robot Manipulation. In *CoRL*.

Gao, C.; Liu, Z.; Chi, Z.; Huang, J.; Fei, X.; Hou, Y.; Zhang, Y.; Lin, Y.; Fang, Z.; Jiang, Z.; and Shao, L. 2025. VLA-OS: Structuring and Dissecting Planning Representations and Paradigms in Vision-Language-Action Models. In *ArXiv*.

Hu, E. J.; Shen, Y.; Wallis, P.; Allen-Zhu, Z.; Li, Y.; Wang, S.; Wang, L.; and Chen, W. 2022. LoRA: Low-Rank Adaptation of Large Language Models. In *ICLR*.

Hu, Y.; Guo, Y.; Wang, P.; Chen, X.; Wang, Y.; Zhang, J.; Sreenath, K.; Lu, C.; and Chen, J. 2025. Video Prediction Policy: A Generalist Robot Policy with Predictive Visual Representations. In *ICML*.

Intelligence, P. 2025a. $\pi 0$: A Vision-Language-Action Flow Model for General Robot Control. In *RSS*.

Intelligence, P. 2025b. $\pi 0.5$: a Vision-Language-Action Model with Open-World Generalization. In *ArXiv*.

Karamcheti, S.; Nair, S.; Balakrishna, A.; Liang, P.; Kollar, T.; and Sadigh, D. 2024. Prismatic VLMs: Investigating the Design Space of Visually-Conditioned Language Models. In *ICML*.

Khazatsky, A.; Pertsch, K.; Nair, S.; Balakrishna, A.; Dasari, S.; Karamcheti, S.; Nasiriany, S.; Srirama, M. K.; Chen, L. Y.; Ellis, K.; Fagan, P. D.; Hejna, J.; Itkina, M.; Lepert, M.; Ma, Y. J.; Miller, P. T.; Wu, J.; Belkhale, S.; Dass, S.; Ha, H.; Jain, A.; Lee, A.; Lee, Y.; Memmel, M.; Park, S.; Radosavovic, I.; Wang, K.; Zhan, A.; Black, K.; Chi, C.; Hatch, K. B.; Lin, S.; Lu, J.; Mercat, J.; Rehman, A.; Sanketi, P. R.; Sharma, A.; Simpson, C.; Vuong, Q.; Walke, H. R.; Wulfe, B.; Xiao, T.; Yang, J. H.; Yavary, A.; Zhao, T. Z.; Agia, C.; Baijal, R.; Castro, M. G.; Chen, D.; Chen, Q.; Chung, T.; Drake, J.; Foster, E. P.; Gao, J.; Guizilini, V.; Herrera, D. A.; Heo, M.; Hsu, K.; Hu, J.; Irshad, M. Z.; Jackson, D.; Le, C.; Li, Y.; Lin, K.; Lin, R.; Ma, Z.; Maddukuri, A.; and Mirchandani, S. 2024. DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset. In *RSS*.

Kim, M. J.; Finn, C.; and Liang, P. 2025. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *RSS*.

Kim, M. J.; Pertsch, K.; Karamcheti, S.; Xiao, T.; Balakrishna, A.; Nair, S.; Rafailov, R.; Foster, E.; Lam, G.; Sanketi, P.; Vuong, Q.; Kollar, T.; Burchfiel, B.; Tedrake, R.; Sadigh, D.; Levine, S.; Liang, P.; and Finn, C. 2024. OpenVLA: An Open-Source Vision-Language-Action Model. In *CoRL*.

Li, X.; Li, P.; Liu, M.; Wang, D.; Liu, J.; Kang, B.; Ma, X.; Kong, T.; Zhang, H.; and Liu, H. 2024a. Towards Generalist Robot Policies: What Matters in Building Vision-Language-Action Models. In *ArXiv*.

Li, X.; Liu, M.; Zhang, H.; Yu, C.; Xu, J.; Wu, H.; Cheang, C.; Jing, Y.; Zhang, W.; Liu, H.; Li, H.; and Kong, T. 2024b. Vision-Language Foundation Models as Effective Robot Imitators. In *ICLR*.

Li, Z.; Chen, G.; Liu, S.; Wang, S.; VS, V.; Ji, Y.; Lan, S.; Zhang, H.; Zhao, Y.; Radhakrishnan, S.; Chang, N.; Sapra, K.; Deshmukh, A. S.; Rintamaki, T.; Le, M.; Karmanov, I.; Voegtle, L.; Fischer, P.; Huang, D.-A.; Roman, T.; Lu, T.; Alvarez, J. M.; Catanzaro, B.; Kautz, J.; Tao, A.; Liu, G.; and Yu, Z. 2025. Eagle 2: Building Post-Training Data Strategies from Scratch for Frontier Vision-Language Models. In *ArXiv*.

Liu, B.; Zhu, Y.; Gao, C.; Feng, Y.; Liu, Q.; Zhu, Y.; and Stone, P. 2023. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *NeurIPS*.

Liu, S.; Wu, L.; Li, B.; Tan, H.; Chen, H.; Wang, Z.; Xu, K.; Su, H.; and Zhu, J. 2025. RDT-1B: A Diffusion Foundation Model for Bimanual Manipulation. In *ICLR*.

Mees, O.; Hermann, L.; Rosete-Beas, E.; and Burgard, W. 2022. CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks. In *IEEE RA-L*.

NVIDIA; Bjorck, J.; Castañeda, F.; Cherniadev, N.; Da, X.; Ding, R.; Fan, L. J.; Fang, Y.; Fox, D.; Hu, F.; Huang, S.; Jang, J.; Jiang, Z.; Kautz, J.; Kundalia, K.; Lao, L.; Li, Z.; Lin, Z.; Lin, K.; Liu, G.; Llontop, E.; Magne, L.; Mandlekar, A.; Narayan, A.; Nasiriany, S.; Reed, S.; Tan, Y. L.; Wang, G.; Wang, Z.; Wang, J.; Wang, Q.; Xiang, J.; Xie, Y.; Xu, Y.; Xu, Z.; Ye, S.; Yu, Z.; Zhang, A.; Zhang, H.; Zhao, Y.;

Zheng, R.; and Zhu, Y. 2025. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots. In *ArXiv*.

Oquab, M.; Darcet, T.; Moutakanni, T.; Vo, H.; Szafraniec, M.; Khalidov, V.; Fernandez, P.; Haziza, D.; Massa, F.; El-Nouby, A.; Assran, M.; Ballas, N.; Galuba, W.; Howes, R.; Huang, P.; Li, S.; Misra, I.; Rabbat, M.; Sharma, V.; Synnaeve, G.; Xu, H.; Jegou, H.; Mairal, J.; Labatut, P.; Joulin, A.; and Bojanowski, P. 2024. DINOv2: Learning Robust Visual Features without Supervision. In *TMLR*.

Peebles, W.; and Xie, S. 2023. Scalable diffusion models with transformers. In *CVPR*.

Pertsch, K.; Stachowicz, K.; Ichter, B.; Driess, D.; Nair, S.; Vuong, Q.; Mees, O.; Finn, C.; and Levine, S. 2025. FAST: Efficient Action Tokenization for Vision-Language-Action Models. In *RSS*.

Qu, D.; Song, H.; Chen, Q.; Yao, Y.; Ye, X.; Ding, Y.; Wang, Z.; Gu, J.; Zhao, B.; Wang, D.; and Li, X. 2025. SpatialVLA: Exploring Spatial Representations for Visual-Language-Action Model. In *ArXiv*.

Reuss, M.; Pari, J.; Agrawal, P.; and Lioutikov, R. 2025. Efficient Diffusion Transformer Policies with Mixture of Expert Denoisers for Multitask Learning. In *ICLR*.

Shentu, Y.; Wu, P.; Rajeswaran, A.; and Abbeel, P. 2024. From LLMs to Actions: Latent Codes as Bridges in Hierarchical Robot Control. In *IROS*.

Shukor, M.; Aubakirova, D.; Capuano, F.; Kooijmans, P.; Palma, S.; Adil Zouitine, M. A.; Pascal, C.; Russi, M.; Marafioti, A.; Alibert, S.; Cord, M.; Wolf, T.; and Cadene, R. 2025. SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics. In *ArXiv*.

Song, H.; Qu, D.; Yao, Y.; Chen, Q.; Lv, Q.; Tang, Y.; Shi, M.; Ren, G.; Yao, M.; Zhao, B.; Wang, D.; and Li, X. 2025a. Hume: Introducing System-2 Thinking in Visual-Language-Action Model. In *ArXiv*.

Song, W.; Chen, J.; Ding, P.; Zhao, H.; Zhao, W.; Zhong, Z.; Ge, Z.; Ma, J.; and Li, H. 2025b. Accelerating Vision-Language-Action Model Integrated with Action Chunking via Parallel Decoding. In *ArXiv*.

Steiner, A.; Pinto, A. S.; Tschannen, M.; Keysers, D.; Wang, X.; Bitton, Y.; Gritsenko, A.; Minderer, M.; Sherbondy, A.; Long, S.; Qin, S.; Ingle, R.; Bugliarello, E.; Kazemzadeh, S.; Mesnard, T.; Alabdulmohsin, I.; Beyer, L.; and Zhai, X. 2024. PaliGemma 2: A Family of Versatile VLMs for Transfer. In *ArXiv*.

Team, O. M.; Ghosh, D.; Walke, H.; Pertsch, K.; Black, K.; Mees, O.; Dasari, S.; Hejna, J.; Kreiman, T.; Xu, C.; Luo, J.; Tan, Y. L.; Chen, L. Y.; Sanketi, P.; Vuong, Q.; Xiao, T.; Sadigh, D.; Finn, C.; and Levine, S. 2024. Octo: An Open-Source Generalist Robot Policy. In *RSS*.

Tian, Y.; Yang, S.; Zeng, J.; Wang, P.; Lin, D.; Dong, H.; and Pang, J. 2025. Predictive Inverse Dynamics Models are Scalable Learners for Robotic Manipulation. In *ICLR*.

Tong, X.; Ding, P.; Fan, Y.; Wang, D.; Zhang, W.; Cui, C.; Sun, M.; Zhao, H.; Zhang, H.; Dang, Y.; Huang, S.; and Lyu, S. 2025. QUART-Online: Latency-Free Large Multimodal Language Model for Quadruped Robot Learning. In *ICRA*.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; Bikel, D.; Blecher, L.; Ferrer, C. C.; Chen, M.; Cucurull, G.; Esiobu, D.; and Fernandes, J. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. In *ArXiv*.

Wang, Y.; Li, X.; Wang, W.; Zhang, J.; Li, Y.; Chen, Y.; Wang, X.; and Zhang, Z. 2025. Unified Vision-Language-Action Model. In *ArXiv*.

Yang, A.; Yang, B.; Hui, B.; Zheng, B.; Yu, B.; Zhou, C.; Li, C.; Li, C.; Liu, D.; Huang, F.; Dong, G.; Wei, H.; Lin, H.; Tang, J.; Wang, J.; Yang, J.; Tu, J.; Zhang, J.; Ma, J.; Yang, J.; Xu, J.; Zhou, J.; Bai, J.; He, J.; Lin, J.; Dang, K.; Lu, K.; Chen, K.; Yang, K.; Li, M.; Xue, M.; Ni, N.; Zhang, P.; Wang, P.; Peng, R.; Men, R.; Gao, R.; Lin, R.; Wang, S.; Bai, S.; Tan, S.; Zhu, T.; Li, T.; Liu, T.; Ge, W.; Deng, X.; Zhou, X.; Ren, X.; Zhang, X.; Wei, X.; Ren, X.; Liu, X.; Fan, Y.; Yao, Y.; Zhang, Y.; Wan, Y.; Chu, Y.; Liu, Y.; Cui, Z.; Zhang, Z.; Guo, Z.; and Fan, Z. 2024. Qwen2 Technical Report. In *ArXiv*.

Yue, Y.; Wang, Y.; Kang, B.; Han, Y.; Wang, S.; Song, S.; Feng, J.; and Huang, G. 2024. DeeR-VLA: Dynamic Inference of Multimodal Large Language Models for Efficient Robot Execution. In *NeurIPS*.

Zhai, X.; Mustafa, B.; Kolesnikov, A.; and Beyer, L. 2023. Sigmoid Loss for Language Image Pre-Training. In *ICCV*.

Zhang, J.; Guo, Y.; Chen, X.; Wang, Y.-J.; Hu, Y.; Shi, C.; and Chen, J. 2024a. HiRT: Enhancing Robotic Control with Hierarchical Robot Transformers. In *CoRL*.

Zhang, R.; Han, J.; Liu, C.; Zhou, A.; Lu, P.; Qiao, Y.; Li, H.; and Gao, P. 2024b. LLaMA-Adapter: Efficient Fine-tuning of Language Models with Zero-init Attention. In *ICLR*.

Zhang, W.; Liu, H.; Qi, Z.; Wang, Y.; Yu, X.; Zhang, J.; Dong, R.; He, J.; Lu, F.; Wang, H.; Zhang, Z.; Yi, L.; Zeng, W.; and Jin, X. 2025. DreamVLA: A Vision-Language-Action Model Dreamed with Comprehensive World Knowledge. In *ArXiv*.

Zhao, Q.; Lu, Y.; Kim, M. J.; Fu, Z.; Zhang, Z.; Wu, Y.; Li, Z.; Ma, Q.; Han, S.; Finn, C.; Handa, A.; Liu, M.-Y.; Xiang, D.; Wetzstein, G.; and Lin, T.-Y. 2025a. SpatialVLA: Exploring Spatial Representations for Visual-Language-Action Model. In *CVPR*.

Zhao, T. Z.; Kumar, V.; Levine, S.; and Finn, C. 2023. Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware. In *RSS*.

Zhao, W.; Ding, P.; Zhang, M.; Gong, Z.; Bai, S.; Zhao, H.; and Wang, D. 2025b. VLAS: Vision-language-action model with speech instructions for customized robot manipulation. In *ICLR*.

Zhong, Z.; Yan, H.; Li, J.; Liu, X.; Gong, X.; Song, W.; Chen, J.; and Li, H. 2025. FlowVLA: Thinking in Motion with a Visual Chain of Thought. In *ArXiv*.