VLAFlow: A Unified Training Framework for Vision-Language-Action Models via Co-training and Future Latent Alignment

Guoyang Xia[1,2,*] Fengfa Li[1,*] Hongjin Ji[1,3] Lei Ren[1,†,‡] Fangxiang Feng[2,‡] Kun Zhan[1]
Yan Xie[1]

[1]Li Auto Inc. [2]School of Artificial Intelligence, Beijing University of Posts and Telecommunications
[3]The Chinese University of Hong Kong, Shenzhen

[*]Equal contribution. [†]Project leader. [‡]Corresponding author.

Abstract

Vision-language-action models (VLAs) have recently advanced robotic manipulation, yet the effects of different robot-data pre-training paradigms remain difficult to compare because existing models often differ in architecture, data, action space, and evaluation protocol. We present **VLAFlow** (**Vision-Language-Action Flow**), a unified flow-matching framework for controlled comparison of VLA training objectives. Using a heterogeneous robot corpus, **OXEMix**, containing approximately $5{,}000$ hours of data from DROID, OpenX-Embodiment, OpenX-Augmented, and RoboCOIN, we evaluate four paradigms under the same π_0-style architecture, shared VLM backbone, action expert, and 14-dimensional action space: action-only modeling (**MindPI**), language-supervised co-training (**MindLPI**), future latent alignment (**MindWPI**), and their combination (**MindLWPI**). Experiments on LIBERO, LIBERO-Plus, and SimplerEnv show that action-only pre-training is sensitive to heterogeneous data. In contrast, language supervision helps preserve vision-language generalization, while future latent alignment improves state-transition and action-outcome modeling. By combining both signals, MindLWPI achieves the most stable overall transfer performance across benchmarks. These results suggest a *meta-action space* view: language and future latent representations provide complementary intermediate constraints that make heterogeneous action supervision smoother and more transferable.

Keywords: vision-language-action models; pre-training paradigms; flow matching; vlm co-training; future latent prediction; robotic manipulation.

Code: https://github.com/MindVLA-Team/VLAFlow

Project Page: https://mindvla-team.github.io/VLAFlow

1 Introduction

Vision-language-action models (VLAs) have recently made rapid progress in robot learning and embodied intelligence. With the development of large-scale robot data, vision-language models, and continuous action generation methods, increasingly many pre-trained VLA foundation models have demonstrated strong cross-task generalization. For example, models such as LingBot-VLA and ABot-M0 leverage large quantities of high-quality robot data for pre-training and achieve significant performance gains on multiple manipulation tasks, providing partial evidence for the scaling potential of VLA models [45, 46]. However, as both data and model scale continue to grow, the cost of pre-training also rises rapidly. How to design more effective training paradigms that can fully exploit heterogeneous robot data and transfer stably to downstream tasks remains insufficiently understood.

Existing studies suggest that VLA pre-training does not yield stable gains simply by increasing data scale. The conventional action-modeling paradigm represented by π_0 usually predicts action chunks directly from visual observations and language instructions, and uses the same objective during downstream fine-tuning [6]. Under such action-only modeling, however, the pre-training data distribution often plays a decisive role in downstream performance. When the pre-training data and downstream task differ substantially in robot embodiment, action space, sampling frequency, or task semantics, the model may suffer negative transfer. This phenomenon indicates that the key to VLA pre-training lies not only in data scale, but also in whether the pre-training objective can learn transferable intermediate representations from highly heterogeneous data [12, 14].

Based on this observation, we focus on a central question: under similar data distributions and the same model architecture, **how do different VLA training paradigms affect downstream transfer performance?** To answer this question, we categorize representative VLA training methods into four groups. The first is the action-only modeling paradigm, represented by π_0, where both pre-training and downstream fine-tuning use action-chunk prediction as the main objective [6]. The second is the VLM co-training paradigm, such as $\pi_{0.5}$, LAP, and JoyAI-RA 0.1, where the VLM is additionally asked to generate subtask goals, action descriptions, or discretized action expressions, thereby introducing high-level action-intent supervision through language space [7, 19, 24]. The third is the future latent feature alignment paradigm, such as Being-H0.7 and LDA-1B, where the model predicts or aligns the latent representation of future frames while predicting actions, enabling feature-level future imagination [18, 30]. The fourth is the combined paradigm explored in this report, where language action descriptions and future latent supervision are introduced simultaneously so that language space and visual latent space jointly constrain action representation learning.

To fill the gap in controlled comparison, we propose **VLAFlow**, or **Vision-Language-Action Flow**. VLAFlow does not refer to a single model variant; rather, it is a unified evaluation framework for VLA training paradigms. "Flow" refers both to the flow-matching action-modeling mechanism in the shared architecture and to how different supervision signals—low-level actions, language intent, and future latent states—flow into the same VLA action-generation framework and ultimately affect downstream robotic control. By placing different pre-training objectives under the same architecture, action space, and

evaluation protocol, VLAFlow enables a more direct analysis of how the training paradigm itself affects transfer performance.

To reduce the experimental cost of large-scale pre-training while maintaining representative data diversity, we construct OXEMix, a medium-scale VLA pre-training corpus of approximately $5{,}000$ hours. The corpus is built from widely used open-source robot datasets, including DROID, OpenX-Embodiment, OpenX-Augmented, and RoboCOIN [17, 20, 32, 44]. In model design, we adopt a π_0-style architecture consistent with recent mainstream approaches and model continuous actions using flow-matching loss, thereby controlling as many confounding factors from architecture and optimization as possible [6, 28]. On this basis, we design and compare four training paradigms: **MindPI**, **MindLPI**, **MindWPI**, and **MindLWPI**.

Specifically, MindPI corresponds to the action-only modeling paradigm and primarily optimizes a continuous-action flow-matching loss during pre-training. MindLPI corresponds to the VLM co-training paradigm and introduces LAP-style [24] action description templates during pre-training, allowing the model to learn both language-level action-intent expression and action modeling; during downstream fine-tuning, we retain only the action-modeling objective to avoid lowering the control frequency with long language generation. MindWPI corresponds to the future latent feature alignment paradigm: it uses V-JEPA 2 as a frozen latent feature extractor, feeds current-frame latent features as additional context to the action expert, and requires the model to predict the latent representation of future frames while still using flow matching for action modeling [2]. MindLWPI further combines MindLPI and MindWPI by using action loss, language action-description loss, and future latent loss during pre-training, and applies average pooling to compress visual latent tokens, reducing token overhead during joint training and inference.

We conduct systematic evaluations on the downstream benchmarks LIBERO, LIBERO-Plus, and SimplerEnv, together with extensive ablations to analyze the mechanisms behind different training paradigms [25, 26, 29]. LIBERO is already close to saturation and thus better serves as a sanity check for basic capability. LIBERO-Plus zero-shot perturbation evaluation reveals the trade-off between the degree of VLM intervention and vision-language generalization. SimplerEnv further amplifies cross-embodiment transfer differences caused by different pre-training objectives. Overall, action-only full-parameter pre-training tends to produce negative transfer on heterogeneous data; language supervision and future latent supervision mitigate this issue from the perspectives of high-level intent and state transition, respectively; and MindLWPI integrates the two to achieve the most stable overall performance across LIBERO, LIBERO-Plus, and SimplerEnv.

We argue that MindLPI, MindWPI, and MindLWPI outperform action-only modeling because they provide intermediate representations for action learning. In highly heterogeneous pre-training data, robot platforms, action definitions, sampling frequencies, and task semantics differ substantially. Relying only on low-dimensional action supervision makes it difficult to form a stable "meta-action" representation. In contrast, language descriptions explicitly express high-level action intent, while future latent features capture action-induced state changes. When combined, these two forms of supervision better smooth the optimization space between action intent and real control signals, helping the model form more generalizable action representations.

The main contributions of this report are summarized as follows:

1. **We propose VLAFlow, a unified flow-matching framework for controlled comparison of VLA training paradigms.** Using approximately $5{,}000$ hours of medium-scale heterogeneous robot data, we conduct a fair comparison of action-only modeling, language-supervised co-training, future latent alignment, and their combination under a unified model architecture, action space, and evaluation protocol.

2. **We show that action-only modeling is sensitive to pre-training settings on heterogeneous robot data and may incur negative transfer.** Experiments indicate that full-parameter action-only pre-training can damage transfer ability on downstream tasks with large distribution gaps. Freezing the VLM can partially preserve vision-language generalization, but does not fully exploit robot data to learn action-related state changes.

3. **We validate the complementarity between language supervision and future latent alignment.** MindLPI provides high-level action-intent supervision through language action descriptions, MindWPI introduces state-transition constraints through future visual latent prediction, and MindLWPI combines the two to obtain more stable transfer performance across multiple benchmarks.

4. **We introduce a meta-action-space interpretation.** Language space and future visual latent space provide intermediate constraints for heterogeneous action supervision from the perspectives of high-level intent and state transition, helping explain transfer differences among VLA training paradigms on heterogeneous robot data.

2 Related Work

2.1 Vision-Language-Action Models and Training Paradigms

Vision-language-action (VLA) models aim to unify visual perception, language understanding, and robot control within an end-to-end framework. Early works such as RT-1 and RT-2 demonstrated the effectiveness of large-scale Transformer-based robot policy learning and the potential for transferring vision-language pre-training knowledge to robotic control [8, 48]. Later, open-source models such as OpenVLA and Octo further advanced VLA architectures and training paradigms: OpenVLA builds a general VLA baseline on Open X-Embodiment using a large language model and visual encoders, whereas Octo emphasizes a lightweight Transformer and a diffusion policy head for efficient fine-tuning and edge deployment [21, 32, 40]. More recently, π_0 introduced flow matching into continuous action generation and combined it with a VLM backbone to enable high-frequency and fine-grained robot control, becoming an important representative VLA architecture [6]. $\pi_{0.5}$ represents a VLM co-training paradigm that goes beyond action imitation by incorporating heterogeneous supervision sources, including high-level semantic subtask prediction, verbal instructions, cross-embodiment robot data, and web multimodal data. Its two-stage recipe first uses FAST-tokenized discrete actions for scalable pre-training, and then adds a flow-matching action expert in post-training to recover fine-grained continuous control [7].

Existing VLA training paradigms can be broadly divided into three categories, and

can be further extended to combined paradigms. The first is the action-only modeling paradigm represented by π_0, whose core objective is to predict action chunks conditioned on visual observations and task instructions [6]. This paradigm has a simple objective and can be scaled to large heterogeneous datasets, but it is also highly dependent on the pre-training data distribution. When the pre-training data differ substantially from the downstream task, action-only modeling can suffer negative transfer. The second is the VLM co-training paradigm. These methods introduce intermediate supervision in language space in addition to action modeling, such as subtask goals, action descriptions, or discretized action tokens. Works such as $\pi_{0.5}$, LAP, and JoyAI-RA 0.1 all exploit language representations to improve modeling of task semantics and action intent to different degrees [7, 19, 24]. The third is the future latent feature alignment paradigm. These methods require the model to predict or align representations of future observations in latent space while predicting actions, giving VLA models feature-level future imagination capability [18, 30, 31, 38]. The MindLWPI paradigm in this report can be viewed as a combination of language-supervised co-training and future latent alignment, designed to study whether the two intermediate supervision signals are complementary.

The development of these training paradigms relies heavily on large-scale robot datasets. Open X-Embodiment aggregates large-scale demonstration trajectories across many robot platforms and is currently one of the most widely used cross-embodiment pre-training corpora [32]. DROID collects diverse manipulation data in real homes and offices [20]. RoboCOIN and RoboMIND further emphasize cross-platform, cross-task, and multi-embodiment data integration [43, 44]. These datasets provide rich data foundations for VLA pre-training, but their high heterogeneity in sampling frequency, action space, task semantics, and robot embodiment also makes the design of effective pre-training objectives a key challenge.

2.2 World Models and Future Prediction in Robot Learning

World models aim to learn environment dynamics so that an agent can internally predict the future outcomes of actions and use such predictions to support decision making. In robot learning, early methods often adopt a decoupled "predict-then-act" framework. For example, UniPi and VidMan predict future observations through video generation models and then recover actions using inverse dynamics models [13, 42]. Later works such as Cosmos Policy and DreamZero attempt to unify future visual prediction and action modeling within a shared generative framework [23, 41]. However, pixel-level prediction typically incurs high computational cost and is susceptible to redundant visual details and long-horizon error accumulation.

Joint-Embedding Predictive Architecture (JEPA) provides a more efficient alternative: instead of reconstructing pixels directly, it predicts features of future or masked regions in representation space [1]. V-JEPA extends this idea to video representation learning, while V-JEPA 2 further demonstrates the potential of scaled self-supervised video pre-training and lightweight robot-data post-training for robot control [2, 4]. Recent VLA-JEPA and JEPA-VLA works introduce the predictive representations of V-JEPA 2 into VLA models, suggesting that future visual latent representations can provide useful dynamics priors for policy learning [31, 38].

Compared with explicit video prediction, latent-space world modeling is better suited for VLA pre-training. Being-H0.7 uses alignment between a prior branch and a posterior branch in latent space, allowing the model to obtain future-aware implicit reasoning at inference time using only current observations [30]. LDA-1B unifies visual forecasting and action generation in a DINO latent space to improve the use of heterogeneous embodied data [18]. Together, these works indicate that future latent prediction can improve the model's understanding of environment dynamics and action outcomes while maintaining inference efficiency. Building on this line of work, this report systematically compares different VLA training paradigms and validates the contribution of future latent alignment, as well as its combination with language supervision, under controlled data and architecture settings.

3 Method

To fairly evaluate the effect of training objectives on downstream VLA transfer, we construct a unified VLAFlow framework and instantiate four training paradigms within it: MindPI, MindLPI, MindWPI, and MindLWPI. Rather than proposing a single new architecture, the central goal of this report is to control architecture, data, action space, and optimization settings so that the main differences among methods are concentrated in the training supervision signals. This section first summarizes the controlled comparison protocol, and then introduces the shared VLAFlow architecture and the four training paradigms. Additional implementation details, including key-value cache sharing, attention masks, action verbalization rules, latent compression, training hyperparameters, and LoRA configurations, are provided in Appendices A–E.

3.1 Controlled Comparison Protocol

This report asks the following question: under approximately the same data distribution and model architecture, do different VLA training objectives lead to different downstream transfer behaviors? To answer this, we adopt four control principles.

Same model backbone. All four paradigms use the same vision-language model (VLM) as the multimodal context encoder, followed by a π_0-style continuous action expert. The action expert models future action chunks with flow matching and does not use an additional discrete action decoder as the downstream control path.

Same action space. All pre-training data are mapped into a 14-dimensional action space. Each arm contributes a 7-dimensional action composed of end-effector translation increments, rotation increments, and gripper state; two arms are concatenated into 14 dimensions. Single-arm data are incorporated into the unified space by zero-padding the unused arm and applying an action-validity mask.

Same data sources and training budget. The four paradigms share OXEMix, a medium-scale open-source robot-data mixture of approximately $5{,}000$ hours, whose main sources include DROID, OpenX-Embodiment, OpenX-Augmented, and RoboCOIN. Except for data fields required by specific auxiliary supervision, all paradigms use the same data sampling strategy, number of training steps, optimizer, and learning-rate configuration.

Table 1: Controlled comparison of four VLA training paradigms.

Paradigm	Auxiliary supervision	PT loss	FT loss	Main role
MindPI	-	$\mathcal{L}_{\text{act}}$	$\mathcal{L}_{\text{act}}$	Action-only transfer baseline.
MindLPI	language	$\mathcal{L}_{\text{act}}, \mathcal{L}_{\text{lang}}$	$\mathcal{L}_{\text{act}}$	Injects high-level action intent through language supervision.
MindWPI	future latent	$\mathcal{L}_{\text{act}}, \mathcal{L}_{\text{lat}}$	$\mathcal{L}_{\text{act}}, \mathcal{L}_{\text{lat}}$	Regularizes learning with future-state prediction.
MindLWPI	language + future latent	$\mathcal{L}_{\text{act}}, \mathcal{L}_{\text{lat}}, \mathcal{L}_{\text{lang}}$	$\mathcal{L}_{\text{act}}, \mathcal{L}_{\text{lat}}$	Combines intent and state-transition constraints.

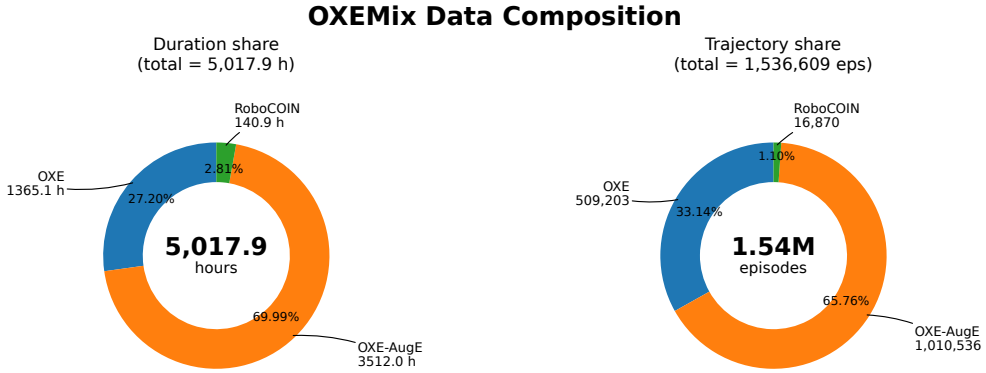

Figure 1: Composition of the OXEMix pre-training corpus in terms of duration and trajectory counts.

Same downstream evaluation protocol. All pre-trained checkpoints are fine-tuned and evaluated on the same downstream benchmarks, including LIBERO, LIBERO-Plus, and SimplerEnv. We also train downstream models initialized without robot-data pre-training as baselines, in order to measure positive or negative transfer induced by different training paradigms.

Table 1 summarizes the main differences among the four paradigms. They share the same inputs, action expert, and downstream control form; the only differences are whether language supervision, future latent supervision, or both are introduced during training.

3.2 Shared VLAFlow Architecture

VLAFlow uses a two-part structure: a VLM backbone plus an action expert. Given multi-view image observations and a language instruction, the VLM first encodes multimodal context; the action expert then generates a continuous action chunk of length T conditioned on this context. We set $T = 16$ by default. To avoid confounding from architectural differences, MindPI, MindLPI, MindWPI, and MindLWPI all use the same shared architecture.

VLM context encoding. The VLM receives images and the language instruction and outputs multi-layer multimodal representations. In our implementation, we use Qwen3-VL-4B-Instruct as the VLM backbone [3]. The action expert does not re-encode images;

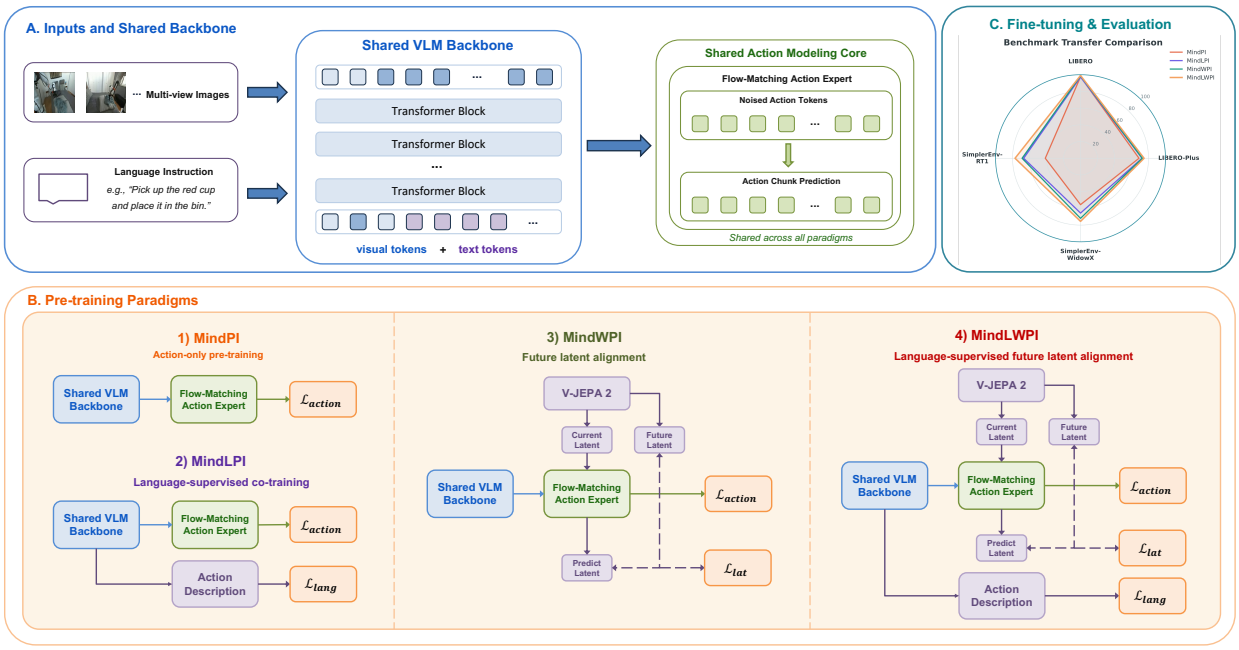

Figure 2: Overview of VLAFlow. VLAFlow conducts controlled comparison of different VLA training paradigms under a unified architecture, data mixture, and evaluation setting. **A. Inputs and shared backbone.** The model takes multi-view visual observations and a language instruction as input, encodes visual and text tokens with a shared VLM backbone, and passes multimodal context to a flow-matching action expert through KV-cache sharing for continuous action-chunk prediction. **B. Pre-training paradigms.** The main diagram shows three basic paradigms (MindPI, MindLPI, MindWPI) and their combination MindLWPI: MindPI uses only action-modeling loss; MindLPI introduces language action-description supervision in addition to the action loss; MindWPI introduces future latent alignment by using V-JEPA 2 to extract current-frame and future-frame latent features and requiring the model to predict future latent representations. MindLWPI, the combined extension proposed in this report, integrates MindLPI and MindWPI; its structure and losses are described in the main text. **C. Fine-tuning and evaluation.** All pre-trained models are fine-tuned under the same downstream protocol and evaluated on LIBERO, LIBERO-Plus, and SimplerEnv.

instead, it reuses the multi-layer context representations from the VLM as conditioning for action generation.

Continuous action expert. The action expert is a diffusion Transformer (DiT) decoder that predicts the flow-matching velocity field for future action chunks [33]. Unlike designs that independently read VLM representations through cross-attention, our implementation uses layer-wise key-value cache sharing, allowing the action expert to reuse multimodal context from different depths of the VLM. This design is shared by all four paradigms and is not a comparison variable; the full formulation is given in Appendix A.2.

Flow-matching action modeling. Let the ground-truth action chunk be $\mathbf{a}$, the noise be $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and the continuous time be $t \in [0, 1]$. The noised action is defined as

$$\mathbf{x}_t = (1 - t)\boldsymbol{\epsilon} + t\mathbf{a}. \tag{1}$$

The action expert predicts a velocity field $\mathbf{v}_\theta(\mathbf{x}_t, t, \mathbf{c})$ conditioned on the VLM context $\mathbf{c}$, with $\mathbf{a} - \boldsymbol{\epsilon}$ as the target:

$$\mathcal{L}_{\text{act}} = \mathbb{E}_{t,\boldsymbol{\epsilon}} \left[\left\| \mathbf{v}_\theta(\mathbf{x}_t, t, \mathbf{c}) - (\mathbf{a} - \boldsymbol{\epsilon}) \right\|^2 \right]. \tag{2}$$

When an action-validity mask is available, the loss is computed only over valid action dimensions. At inference time, the model starts from Gaussian noise and generates action chunks through a small number of Euler integration steps. All paradigms deploy only continuous action denoising and do not perform autoregressive language generation, so auxiliary training tasks do not affect control frequency. This formulation follows the general flow-matching framework and recent VLA flow policies [6, 28].

3.3 MindPI: Action-Only Pre-training

MindPI corresponds to the action-only modeling paradigm represented by π_0. Its training input consists of the current visual observation, language instruction, and noised action chunk; its only supervision signal is the future action chunk. Both pre-training and downstream fine-tuning optimize the flow-matching action loss in Eq. (2):

$$\mathcal{L}_{\text{MindPI}} = \mathcal{L}_{\text{act}}. \tag{3}$$

MindPI is the key baseline of this report. Because it introduces no language description, future state, or other intermediate supervision, the model must rely entirely on low-dimensional action labels to learn transferable representations from heterogeneous robot data. Therefore, MindPI can be used to test the training stability of action-only modeling across robot embodiments, sampling frequencies, and action spaces.

3.4 MindLPI: Language-Supervised VLM Co-training

MindLPI corresponds to the VLM co-training paradigm. Its core idea is to introduce intermediate supervision in language space in addition to action modeling, so that the VLM learns to encode high-level action intent in linguistic representations. Specifically, we convert action chunks into LAP-style action-description text and use it as an autoregressive training target for the VLM [24]. The language supervision in the current experiments is solely derived from action-description templates.

The MindLPI pre-training objective consists of the continuous action loss $\mathcal{L}_{\text{act}}$ and the language action-description loss $\mathcal{L}_{\text{lang}}$:

$$\mathcal{L}_{\text{MindLPI}} = \mathcal{L}_{\text{act}} + \lambda_{\text{lang}}\mathcal{L}_{\text{lang}}. \tag{4}$$

Here $\mathcal{L}_{\text{lang}}$ is the standard autoregressive cross-entropy loss, and we set $\lambda_{\text{lang}} = 0.1$ in the current experiments. Language action descriptions can take the form of discretized integer action sequences or natural-language action templates; details are provided in Appendix B. In ablations, we compare whether the action loss is backpropagated into the VLM, and the results show that removing gradient truncation is more suitable as the main setting.

During downstream fine-tuning, MindLPI keeps only the continuous action loss $\mathcal{L}_{\text{act}}$ and no longer generates language action descriptions. This design matches deployment requirements: language supervision shapes representations only during pre-training, while closed-loop control does not introduce additional autoregressive language latency.

3.5 MindWPI: Future Latent Feature Alignment

MindWPI corresponds to the future latent feature alignment paradigm. Its core hypothesis is that effective VLA pre-training should learn not only a mapping from current observations to actions, but also state changes that actions may induce. Compared with pixel-level future-frame reconstruction, latent-space prediction avoids modeling visual details weakly related to control, such as texture, lighting, and background, while retaining compact information about dynamics, contact, and task progress.

Specifically, we use a frozen V-JEPA 2 model as the latent feature extractor [2]. Given current and future frames, we extract latent representations $\mathbf{z}_{\text{cur}}$ and $\mathbf{z}_{\text{fut}}$. The current latent representation is encoded and used as additional tokens for the action expert; while predicting the action velocity field, the action expert also predicts the future latent representation $\hat{\mathbf{z}}_{\text{fut}}$ through a latent decoder. The future latent loss is defined as

$$\mathcal{L}_{\text{lat}} = \|\hat{\mathbf{z}}_{\text{fut}} - \mathbf{z}_{\text{fut}}\|^2 . \tag{5}$$

The joint objective of MindWPI is

$$\mathcal{L}_{\text{MindWPI}} = \mathcal{L}_{\text{act}} + \lambda_{\text{lat}}\mathcal{L}_{\text{lat}}. \tag{6}$$

To prevent future latent prediction from taking a shortcut through action tokens, we use a structured attention mask: latent tokens can predict the future only from the current observation and language context, and cannot attend to noised action tokens; action tokens, however, can attend to latent tokens and use them as predictive visual context. The full attention-mask form is given in Appendix A.3.

During downstream fine-tuning, MindWPI continues to retain both action loss and future latent loss. At inference time, the model requires only the current-frame latent representation as condition and outputs both the future latent representation and the action chunk. Therefore, MindWPI uses future prediction as a representation constraint during training without changing the deployment interface for action generation.

3.6 MindLWPI: Joint Language Supervision and Future Latent Pre-training

MindLWPI (Language-supervised Future Latent Alignment) further combines MindLPI and MindWPI. The motivation is that language action descriptions provide high-level action-intent supervision, while future latent prediction provides action-outcome and state-transition supervision. The two respectively constrain “what to do” and “what the action will change.” Thus, MindLWPI simultaneously optimizes action modeling, language action description, and future latent prediction during pre-training:

$$\mathcal{L}_{\text{MindLWPI}} = \mathcal{L}_{\text{act}} + \lambda_{\text{lat}}\mathcal{L}_{\text{lat}} + \lambda_{\text{lang}}\mathcal{L}_{\text{lang}}. \tag{7}$$

In all reported MindLWPI results, when both the action loss and future latent loss are enabled, their ratio during pre-training is fixed at $1:1$. The language action-description loss weight follows the MindLPI setting, namely $\lambda_{\text{lang}} = 0.1$. As in MindLPI, MindLWPI disables the language loss during downstream fine-tuning and retains only the action loss and future latent loss.

Directly using all 256 V-JEPA 2 latent tokens for both language co-training and future prediction substantially increases the sequence length of the action expert. To reduce inference overhead, MindLWPI uses AvgPool-k4 compression by default: every 4 adjacent tokens along the latent-token sequence are averaged, reducing 256 tokens to 64 tokens. This compression is applied to both the current latent prefix and the future latent target so that the prediction dimensions are consistent. We compare AvgPool and MLP compression, as well as k4 and k16 settings, in ablation experiments. The results show that simple AvgPool-k4 offers a favorable trade-off between performance and computational cost. Therefore, unless otherwise stated, MindLWPI in the main text refers to "language supervision + future latent alignment + AvgPool-k4 + downstream $\lambda_{lat}^{\text{ft}}/\lambda_{act}^{\text{ft}} = 0.1$."

3.7 Training, Fine-tuning, and Efficient Adaptation Protocol

The four paradigms share the same pre-training data mixture and optimization budget. Pre-training data are uniformly converted into the LeRobot format and mapped into the 14-dimensional action space described in Section 3.1 [10]. To address sampling imbalance caused by different dataset sizes, we use a sampling strategy that balances dataset scale and trajectory length. Optimizer, learning rate, training steps, and flow-matching sampling details are provided in Appendix D.

At the downstream stage, we transfer checkpoints from different paradigms to the same evaluation benchmarks and use the same fine-tuning protocol. Whether a pre-trained checkpoint is loaded distinguishes pre-training transfer results from downstream baselines without pre-training. For benchmarks such as LIBERO whose action definitions differ from the pre-training phase, the VLM mainly transfers vision-language representations, while the action expert adapts to the new action space with a higher learning rate.

In addition to full-parameter fine-tuning, we also explore low-rank adaptation (LoRA) to evaluate downstream adaptation under resource constraints [15]. LoRA experiments are not a primary variable in the comparison of the four training paradigms, so their implementation details and injection locations are provided in Appendix E; the experimental results are reported in Section 4.

4 Experiments

4.1 Experimental Setup

Evaluation benchmarks and metrics. We evaluate different training paradigms on three downstream benchmarks. **LIBERO** contains four standard suites: L-Spatial, L-Object, L-Goal, and L-Long. Each suite is evaluated over 500 rollouts, and task success rate (%) is used as the metric [29]. **LIBERO-Plus** introduces seven types of perturbations on top of standard LIBERO tasks, including camera viewpoint, robot initial state, language instruction, lighting, background texture, sensor noise, and object layout, to evaluate zero-shot robustness [25]. All models are trained or fine-tuned only on standard LIBERO data and are not adapted on LIBERO-Plus. **SimplerEnv** contains two robot platforms, WidowX and RT-1 [26]. WidowX evaluates four tasks: Stack, Carrot, Spoon, and Eggplant; RT-1 evaluates

two visual settings, Visual Matching (VM) and Visual Augmentation (VA). Unless otherwise stated, all results are success rates (%).

Compared methods. To disentangle architectural gains, pre-training gains, and training-objective gains, we compare seven VLAFlow variants: MindPI w/o PT, MindWPI w/o PT, MindPI (Frozen VLM), MindPI (Full PT), MindLPI, MindWPI, and MindLWPI. Here w/o PT means no robot-data pre-training, and Full PT means that both the VLM and action expert are updated during pre-training. MindLWPI by default denotes Language-supervised Future Latent Alignment with AvgPool-k4 latent compression and downstream $\lambda_{lat}^{\text{ft}}/\lambda_{act}^{\text{ft}} = 0.1$.

Checkpoint and downstream-data selection rules. For LIBERO and LIBERO-Plus, we uniformly use the 100k-step checkpoint. For the main SimplerEnv table, we adopt a split fine-tuning strategy: WidowX uses the best checkpoint from Bridge-only fine-tuning, whereas RT-1 uses the best checkpoint from RT-1-only fine-tuning. For many low-cost ablations, we use a mixed SimplerEnv fine-tuning setting with both Bridge and RT-1 data to reduce the cost of full pre-training and per-platform fine-tuning; Section 4.4 discusses this trade-off explicitly.

Definition of negative transfer. We define negative transfer as the case where a model pre-trained on robot data performs worse than its corresponding no-pre-training baseline under the same downstream fine-tuning protocol. This definition is used to analyze whether heterogeneous pre-training data are stably converted into downstream gains under different training paradigms.

4.2 Controlled Comparison: Overall Transfer Behavior Across Training Paradigms

Table 2 summarizes the controlled comparison of major VLAFlow variants on the three downstream benchmarks. This table compares the transfer behavior of different training objectives under the same architecture, action space, and evaluation protocol.

Table 2: Controlled comparison of VLAFlow training paradigms. For SimplerEnv, WidowX uses Bridge-only fine-tuning results, and RT-1 uses RT-1-only fine-tuning results. MindLWPI uses AvgPool-k4 latent compression and downstream $0.1 : 1$ FT ratio by default.

Method	Robot pre-training	Auxiliary supervision	LIBERO Avg	LIBERO-Plus Total	WidowX Avg	RT-1 VM	RT-1 VA
No robot-data pre-training							
MindPI w/o PT	No	-	97.0	59.9	59.6	75.7	60.4
MindWPI w/o PT	No	future latent	97.4	66.1	71.9	75.2	51.6
Action-only pre-training							
MindPI (Frozen VLM)	Yes	-	97.2	**74.9**	54.4	72.7	66.0
MindPI (Full PT)	Yes	-	97.5	68.8	65.9	68.2	55.5
Auxiliary-supervised pre-training							
MindLPI	Yes	language	97.2	72.3	65.6	74.6	59.2
MindWPI	Yes	future latent	98.5	72.6	74.5	**86.7**	**71.1**
MindLWPI	Yes	language + future latent	**99.1**	74.8	**75.5**	84.4	69.8

LIBERO is already close to saturation, with absolute differences mainly within 1–2 percentage points; it is therefore better suited as an in-distribution capability check. In contrast, LIBERO-Plus and SimplerEnv better reveal transfer differences among training

paradigms. MindPI (Full PT) improves over the no-pretraining baseline on WidowX but degrades substantially on RT-1 VM/VA, reflecting the instability and potential negative transfer of action-only pre-training under heterogeneous robot data. Among auxiliary-supervised paradigms, MindWPI achieves the best RT-1 VM/VA performance and remains competitive on LIBERO and WidowX, suggesting that future latent supervision is particularly effective for modeling action outcomes and state transitions. MindLWPI obtains the best results on LIBERO, LIBERO-Plus, and WidowX, while remaining close to the best RT-1 performance. This indicates that language supervision and future latent supervision are complementary, with future latent alignment contributing strongly to cross-platform control transfer and language supervision improving overall robustness across benchmarks.

4.3 Comparison with Public Baselines

We report public baselines following the common format used in recent VLA technical reports. These baselines provide absolute-performance references, whereas VLAFlow internal variants are used for controlled comparison. Representative references include OpenVLA, OpenVLA-OFT, π_0, $\pi_{0.5}$, and GR00T-N1 [5–7, 21, 22].

LIBERO. Table 3 reports LIBERO results in terms of L-Spatial, L-Object, L-Goal, L-Long, and their average. Public baselines provide absolute-performance references, while VLAFlow variants are compared under the unified architecture and evaluation protocol.

Table 3: Public baselines and VLAFlow results on the LIBERO benchmark (success rate, %). External baselines are from public technical reports; VLAFlow variants are compared under a unified architecture and evaluation protocol.

Method	L-Spatial	L-Object	L-Goal	L-Long	Avg
OpenVLA [21]	84.7	88.4	79.2	53.7	76.5
π_0-Fast [34]	96.4	96.8	88.6	60.2	85.5
GROOT-N1 [5]	94.4	97.6	93.0	90.6	93.9
π_0 [6]	98.0	96.8	94.4	88.4	94.4
$\pi_{0.5}$ [7]	98.8	98.2	98.0	92.4	96.9
OpenVLA-OFT [22]	97.6	98.4	97.9	94.5	97.1
MindPI w/o PT	97.8	99.0	97.0	94.0	97.0
MindPI (Frozen VLM)	98.4	99.2	96.8	94.6	97.2
MindPI (Full PT)	98.2	98.4	98.0	95.2	97.5
MindLPI	97.8	98.4	98.0	94.8	97.2
MindWPI w/o PT	98.0	99.6	98.2	93.6	97.4
MindWPI	99.0	99.6	98.6	96.8	98.5
MindLWPI	**99.2**	**99.8**	**99.2**	**98.2**	**99.1**

VLAFlow reaches a level close to recent strong baselines on LIBERO. MindLWPI obtains high scores on all four suites, with a particularly large improvement on L-Long, suggesting that the combination of language supervision and future latent supervision helps long-horizon action chains. However, because LIBERO is nearly saturated overall, the following analysis focuses more on out-of-distribution transfer in LIBERO-Plus and SimplerEnv.

LIBERO-Plus. Table 4 reports zero-shot robustness by official LIBERO-Plus perturbation category. Total is computed using the official aggregation protocol rather than a simple arithmetic mean over the seven perturbation types.

Table 4: LIBERO-Plus zero-shot robustness comparison (success rate, %). All methods are trained only on standard LIBERO and are not fine-tuned on LIBERO-Plus. Total uses the official aggregation protocol rather than a simple average over the seven perturbations.

Method	Camera	Robot	Language	Light	Background	Noise	Layout	Total
OpenVLA [21]	0.8	3.5	23.0	8.1	34.8	15.2	28.5	15.6
OpenVLA-OFT [22]	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
OpenVLA-OFT-w [22]	10.4	38.7	70.5	76.8	93.6	49.9	69.9	55.8
OpenVLA-OFT-m [22]	55.6	21.7	81.0	92.7	91.0	78.6	68.7	67.9
NORA [16]	2.2	37.0	65.1	45.7	58.6	12.8	62.1	39.0
WorldVLA [11]	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.0
UniVLA [9]	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9
π_0 [6]	13.8	6.0	58.8	85.0	81.4	79.0	68.9	53.6
π_0-Fast [34]	65.1	21.6	61.0	73.2	73.2	74.4	68.8	61.6
RIPT-VLA [39]	55.2	31.2	77.6	88.4	91.6	73.5	74.2	68.4
MindPI w/o PT	26.1	32.0	71.0	88.0	94.4	58.3	70.3	59.9
MindWPI w/o PT	32.7	48.7	77.0	90.7	93.2	64.0	73.6	66.1
MindPI (Frozen VLM)	42.3	58.6	86.6	94.7	96.3	81.9	78.0	74.9
MindPI (Full PT)	42.0	57.8	75.4	91.5	93.2	57.4	86.7	68.8
MindLPI	46.2	61.7	84.6	93.4	95.2	58.4	82.2	72.3
MindWPI	36.7	73.8	86.2	96.5	91.6	54.2	84.8	72.6
MindLWPI	35.1	**74.2**	**93.0**	**97.1**	**96.7**	58.9	**84.5**	**74.8**

LIBERO-Plus reveals the different generalization preferences of different supervision signals. MindPI (Frozen VLM) better preserves the generalization capability of the original VLM. MindWPI performs especially well on Robot Initial States and Objects Layout, suggesting that future latent alignment is well suited for modeling state changes and spatial layouts. MindLWPI further improves Language, Light, Background, and Total, indicating complementarity between language supervision and future latent supervision.

SimplerEnv. Table 5 reports SimplerEnv results in the format commonly used in public papers. Since the evaluation protocol is the same, we place VLAFlow variants alongside public baselines.

Table 5: Public baselines and VLAFlow results on SimplerEnv (success rate, %). RT-1 reports Visual Matching (VM) and Visual Augmentation (VA), while WidowX is the average over four tasks. VLAFlow uses Bridge-only fine-tuning for WidowX and RT-1-only fine-tuning for RT-1.

Method	Size	RT-1 VM	RT-1 VA	WidowX
SpatialVLA [35]	4B	75.1	70.7	42.7
FPC-VLA [47]	7B	78.0	65.8	64.6
MemoryVLA [37]	7B	77.7	72.7	71.9
π_0 [6]	3B	58.8	56.8	27.8
π_0+FAST [34]	3B	61.9	60.5	39.5
OpenVLA-OFT [22]	7B	63.0	54.3	31.3
DD-VLA [27]	7B	71.2	64.1	49.3
MindPI w/o PT	4B	75.7	60.4	59.6
MindWPI w/o PT	4B	75.2	51.6	71.9
MindPI (Frozen VLM)	4B	72.7	66.0	54.4
MindPI (Full PT)	4B	68.2	55.5	65.9
MindLPI	4B	74.6	59.2	65.6
MindWPI	4B	**86.7**	**71.1**	74.5
MindLWPI	4B	84.4	69.8	**75.5**

SimplerEnv has a much larger distribution gap than LIBERO and therefore better exposes transfer differences among training objectives. Compared with mixed fine-tuning, platform-specific fine-tuning on Bridge and RT-1 better reflects whether a pre-trained representation can adapt to each target embodiment. MindWPI obtains the strongest RT-1 VM/VA results, showing that future latent alignment provides an effective training signal for modeling action outcomes and state transitions under cross-platform distribution shifts. In contrast, MindLWPI achieves the best WidowX performance and remains close to the best RT-1 scores. These results suggest that language supervision and future latent alignment are complementary, but their benefits are task- and platform-dependent: future latent alignment contributes more strongly to RT-1 transfer, whereas the combined language-and-latent supervision yields the most robust performance on WidowX.

4.4 Mixed and Split Fine-tuning on SimplerEnv: Bias in a Low-Cost Ablation Environment

In early hyperparameter ablations, we used a mixed SimplerEnv fine-tuning setting with both Bridge and RT-1 data to construct a lower-cost and faster-iteration downstream adaptation environment. This setting can provide rapid feedback on the relative trends of design choices such as future latent loss ratio, future-frame offset, and latent routing without requiring full per-platform fine-tuning. However, Bridge/WidowX and RT-1 differ in robot embodiment, visual distribution, action scale, and task composition; therefore, the absolute numbers from mixed fine-tuning may be affected by cross-platform data interference. We further compare mixed Simpler FT and Bridge-only FT on WidowX to calibrate the bias of this low-cost ablation environment.

Table 6: WidowX comparison between mixed SimplerEnv fine-tuning and Bridge-only fine-tuning (success rate, %). This table is only used to quantify the numerical bias of mixed fine-tuning as a low-cost hyperparameter-screening environment, and is not used to compare full pre-training gains.

Method	Mixed Simpler FT	Bridge-only FT	Change
MindPI w/o PT	63.0	59.6	-3.4
MindWPI w/o PT	73.4	71.9	-1.5
MindPI	55.5	65.9	+10.4
MindWPI	71.9	74.5	+2.6

Table 6 shows that mixed fine-tuning and Bridge-only fine-tuning exhibit a clear numerical discrepancy, and this discrepancy becomes more pronounced after loading robot-data pre-training weights. For models without pre-training, mixed data may primarily increase data volume and act as regularization. For pre-trained models, however, the model has already acquired a certain cross-embodiment action prior from OXEMix; applying Bridge and RT-1 simultaneously during downstream adaptation may introduce cross-platform gradient conflicts or compromise optima, thereby weakening adaptation to a specific target platform. This interpretation is consistent with prior observations in multi-task gradient conflict and robot data-mixture optimization [12, 14]. Based on this observation, later ablations that use mixed Simpler FT are used only to compare relative trends among design choices. In the final SimplerEnv main results, we adopt a per-platform fine-tuning

protocol: WidowX uses Bridge-only FT, and RT-1 uses RT-1-only FT.

4.5 Ablation Studies

Gradient truncation in MindLPI. Table 7 compares whether the action loss in MindLPI is backpropagated into the VLM. The initial motivation for gradient truncation was to prevent low-level action supervision from interfering with the VLM, but the results show that this strategy causes substantial degradation on LIBERO-Plus.

Table 7: MindLPI gradient-truncation ablation (success rate, %).

Method	LIBERO Avg	LIBERO-Plus Total
MindLPI w/ stop-gradient	96.3	45.8
MindLPI w/o stop-gradient	**97.2**	**72.3**

Removing gradient truncation is markedly better than using it, indicating that backpropagation from the action loss into the VLM is not merely noise; it may provide action-outcome alignment signals for vision-language representations. Therefore, the main MindLPI experiments use the no-stop-gradient setting.

Downstream fine-tuning loss ratio in MindWPI. Table 8 shows the downstream fine-tuning loss-ratio ablation after full pre-training. It is important to note that in all currently reported full MindWPI results, when both action loss and future latent loss are enabled during pre-training, the pre-training ratio is fixed at $1:1$; the table varies the downstream fine-tuning ratio $\lambda^{\text{ft}}_{lat}/\lambda^{\text{ft}}_{act}$.

Table 8: MindWPI loss-ratio ablation after pre-training on OXEMix and fine-tuning on SimplerEnv WidowX (success rate, %). The pre-training target column indicates the loss composition used during robot-data pre-training, while $\lambda^{\text{ft}}_{\text{lat}}$ and $\lambda^{\text{ft}}_{\text{act}}$ denote downstream fine-tuning loss weights.

Method	Pre-training target	$\lambda^{\text{ft}}_{\text{lat}}$	$\lambda^{\text{ft}}_{\text{act}}$	WidowX Avg
MindWPI (Latent-only PT)	latent only $(0:1)$	0.1	1.0	64.6
MindWPI (1:1 FT)	act+lat $(1:1)$	1.0	1.0	61.5
MindWPI (Action-only FT)	act+lat $(1:1)$	0	1.0	70.8
MindWPI (0.01:1 FT)	act+lat $(1:1)$	0.01	1.0	67.2
MindWPI (1:0.1 PT)	act+lat $(1:0.1)$	0.1	1.0	67.7
MindWPI (1:1 PT)	act+lat $(1:1)$	0.1	1.0	**71.9**

The results show that the future latent objective needs to be balanced carefully in both pre-training and downstream fine-tuning. First, using latent prediction alone during pre-training is insufficient: MindWPI (Latent-only PT) reaches only 64.6, indicating that future visual prediction by itself cannot replace action supervision for downstream control transfer. Second, under the same downstream fine-tuning ratio of $\lambda^{\text{ft}}_{\text{lat}} : \lambda^{\text{ft}}_{\text{act}} = 0.1:1$, the pre-training ratio matters substantially: act+lat $(1:0.1)$ reaches 67.7, whereas act+lat $(1:1)$ improves to 71.9. This suggests that a weak latent loss during pre-training does not provide a sufficiently strong state-transition constraint, while the balanced $1:1$ pre-training objective better couples action modeling with future-state prediction.

The downstream fine-tuning ratio is also important. Starting from the same act+lat $(1:1)$ pre-trained checkpoint, a $1:1$ FT ratio performs poorly, likely because the latent constraint becomes too strong during target-domain adaptation and competes with the action objective. Removing the latent loss during fine-tuning improves performance, but still underperforms the recommended $0.1:1$ FT ratio. Therefore, our default MindWPI recipe uses a balanced action-latent objective during pre-training and a weaker future latent regularizer during downstream fine-tuning, namely act+lat $(1:1)$ for pre-training and $\lambda_{\text{lat}}^{\text{ft}} : \lambda_{\text{act}}^{\text{ft}} = 0.1:1$ for fine-tuning.

Low-cost w/o PT design ablations. Because full pre-training is expensive, we also conduct low-cost ablations of MindWPI design choices without robot-data pre-training. This group of experiments is not used to claim full pre-training gains; rather, it verifies whether future latent loss, future-frame offset, and latent routing design affect downstream adaptation.

Table 9: MindWPI w/o PT design ablation (SimplerEnv WidowX success rate, %). None of the rows uses robot-data pre-training.

Method	Loss setting	Future-frame offset	WidowX Avg
MindWPI w/o PT (1:1 FT)	1:1	8	64.3
MindWPI w/o PT (Action-only FT)	action only	-	69.0
MindWPI w/o PT	0.1:1	8	**73.4**
MindWPI w/o PT (0.1:1, Offset 32)	0.1:1	32	59.9

Without pre-training, downstream $\lambda_{lat}^{\text{ft}}/\lambda_{act}^{\text{ft}} = 0.1$ remains significantly better than the $1:1$ setting, indicating that the latent loss ratio is also sensitive for downstream adaptation itself. Offset 32 is clearly weaker than the default Offset 8, suggesting that a future-frame interval that is too long may introduce state changes weakly associated with the current action.

MindLWPI and latent compression. Table 10 summarizes the results for latent-token compression and joint supervision in MindWPI/MindLWPI. AvgPool-k4 reduces the number of tokens from 256 to 64 while preserving strong performance; k16 and MLP compression are less stable in low-cost ablations. Therefore, the main text uses AvgPool-k4 as the default compression method for MindLWPI.

Table 10: Latent compression and joint-supervision ablation (SimplerEnv WidowX success rate, %). Compression experiments are mainly used to justify AvgPool-k4 as the default setting for MindLWPI.

Setting	Compression method	Training stage	WidowX Avg
MindWPI w/o PT	No compression	Direct downstream fine-tuning	73.4
MindWPI compressed	AvgPool-k4	Direct downstream fine-tuning	74.0
MindWPI compressed	AvgPool-k16	Direct downstream fine-tuning	67.2
MindWPI compressed	MLP-k4	Direct downstream fine-tuning	70.8
MindWPI compressed	MLP-k16	Direct downstream fine-tuning	60.7
MindLWPI	AvgPool-k4	Full pre-training + Bridge FT	**75.5**

Pre-training data composition. Table 11 compares how different robot pre-training data sources affect the MindPI action-only modeling paradigm.

Table 11: MindPI pre-training data-source ablation on SimplerEnv WidowX (success rate, %). OXE raw subset denotes the original OXE subset including DROID, while excluding OXE-Augmented and RoboCOIN.

Pre-training data	Stack	Carrot	Spoon	Eggplant	WidowX Avg
No PT	32.3	58.3	67.7	93.8	63.0
RoboCOIN subset	0.05	31.3	38.5	91.7	40.4
OXE-Augmented	13.5	59.4	84.4	81.3	59.7
DROID only	46.9	52.1	81.2	68.8	62.2
OXE raw subset	37.5	57.3	**87.5**	78.1	**65.1**

This ablation shows that the effect of action-only robot-data pre-training is highly dependent on the source and composition of the pre-training corpus. Different subsets lead to substantially different transfer behaviors: RoboCOIN subset causes severe degradation on WidowX, OXE-Augmented improves some manipulation categories but lowers the overall average, while DROID-only and OXE raw subset provide gains on certain tasks but still introduce task-specific regressions. These results suggest that negative transfer is not caused by robot-data pre-training per se, but by the difficulty of aligning heterogeneous embodiments, action definitions, task distributions, and visual domains under a purely low-dimensional action-supervision objective. This further supports our view that large-scale heterogeneous VLA pre-training requires intermediate constraints beyond action imitation alone in order to form stable and transferable action representations.

4.6 Efficient Adaptation with LoRA

Considering the storage and computational cost of full-parameter fine-tuning, we additionally evaluate low-rank adaptation with LoRA [15]. This experiment inherits the OXEMix pre-training weights from MindPI (Full PT) and is tested only on LIBERO for rapid evaluation; it is not a core variable in the comparison of training paradigms. Figure 3 shows LIBERO average success rate as the number of trainable parameters varies across LoRA injection locations, and Table 12 reports complete results for the four LIBERO suites.

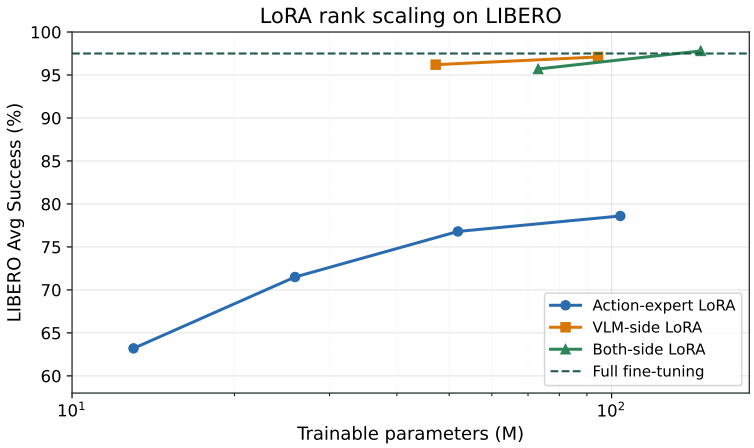

Figure 3: LoRA rank scaling on LIBERO. The horizontal axis is the number of trainable parameters (M), and the vertical axis is the average success rate over the four LIBERO suites. The horizontal dashed line indicates the MindPI (Full PT) full-parameter fine-tuning baseline.

Table 12: Efficient LoRA adaptation results on LIBERO (success rate, %). LIBERO contains four suites: L-Spatial, L-Object, L-Goal, and L-Long. Avg is the average over the four suites.

Method	Trainable parameters	L-Spatial	L-Object	L-Goal	L-Long	Avg
Action LoRA r=64	13.0M	71.8	85.2	55.2	40.6	63.2
Action LoRA r=128	25.9M	83.0	88.8	64.8	49.4	71.5
Action LoRA r=256	51.9M	88.4	93.6	69.2	56.0	76.8
Action LoRA r=512	103.8M	87.8	95.0	68.8	63.0	78.6
VLM LoRA r=128	47.2M	97.4	95.8	99.2	92.2	96.2
VLM LoRA r=256	94.4M	98.2	99.2	98.8	92.2	97.1
Both LoRA r=128	73.1M	96.8	99.0	97.8	89.2	95.7
Both LoRA r=256	146.3M	98.0	99.6	98.8	94.8	**97.8**

The results show that injecting LoRA only into the action expert makes it difficult to approach full-parameter fine-tuning. VLM-side LoRA already approaches full fine-tuning with around 100M trainable parameters, while both-side LoRA achieves the highest average score under a larger parameter budget. This suggests that for VLM-based VLA models, downstream adaptation requires not only updating the action expert, but also providing sufficient low-rank adaptation capacity for vision-language representations.

4.7 Meta-Action Space Analysis

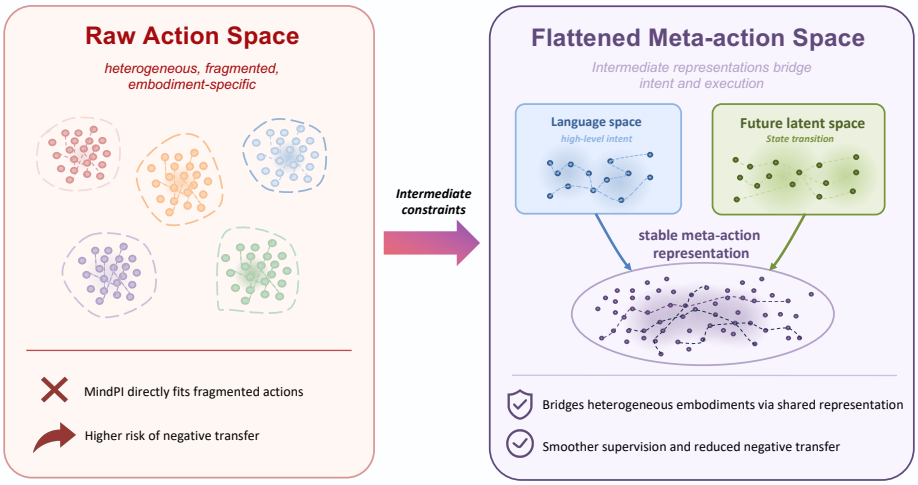

Figure 4: Comparison of training paradigms from the perspective of meta-action space. MindPI directly fits heterogeneous action labels and is vulnerable to differences in robot embodiment, sampling frequency, and action definition. MindLPI provides high-level action-intent constraints through language space. MindWPI provides state-transition constraints through future visual latent space. MindLWPI jointly uses language and future latent space so that action intent and state transition constrain action representations in the same framework, forming a smoother meta-action space.

Taken together, the experimental results show that the four paradigms differ in how they form "meta-action representations." MindPI directly fits heterogeneous action labels and is easily affected by differences in robot embodiment, sampling frequency, and action definition. MindLPI provides high-level action-intent constraints through language space. MindWPI provides state-transition constraints through future visual latent space. These two intermediate supervision signals mitigate the instability of heterogeneous action supervision from the complementary perspectives of semantic intent and action outcome. The results of MindLWPI show that the two are not mutually exclusive: language supervision helps the model organize high-level task intent, while future latent prediction helps the model capture action-induced state changes. Combining them leads to smoother and more transferable action representations.

5 Conclusion

This report presents VLAFlow, a unified flow-matching framework for controlled comparison of VLA training paradigms. Using approximately $5{,}000$ hours of OXEMix heterogeneous robot data, a unified π_0-style architecture, and a unified 14-dimensional action space, we compare four paradigms: action-only modeling (MindPI), language-supervised co-training (MindLPI), future latent alignment (MindWPI), and their combination (MindLWPI). Experiments show that action-only pre-training is sensitive to data distribution and VLM update strategy, leading to unstable transfer under large distribution gaps. Language supervision and future latent supervision provide complementary intermediate constraints from action intent and state transition, respectively. MindWPI achieves the strongest RT-1 transfer

performance, highlighting the value of future latent alignment for cross-platform control. MindLWPI further combines language and future latent supervision, obtaining the best results on LIBERO, LIBERO-Plus, and WidowX while remaining competitive on RT-1. These findings support a meta-action-space view: intermediate language and latent representations help smooth heterogeneous action supervision and improve transferability. Future work will scale pre-training, explore pre-training-stage loss ratios, and validate the framework on real robot platforms.

References

[1] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15619–15629, 2023.

[2] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

[3] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[4] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

[5] 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.

[6] 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. *arXiv preprint arXiv:2410.24164*, 2024.

[7] Kevin Black, Noah Brown, Danny Driess, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization, 2025.

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

[9] 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. *arXiv preprint arXiv:2505.06111*, 2025.

[10] Remi Cadene, Simon Alibert, Alexander Soare, Quentin Gallouedec, Adil Zouitine, Steven Palma, Pepijn Kooijmans, Michel Aractingi, Mustafa Shukor, Dana Aubakirova, et al. Lerobot: State-of-the-art machine learning for real-world robotics in pytorch, 2024.

[11] Jun Cen, Chaohui Yu, Hangjie Yuan, Yuming Jiang, Siteng Huang, Jiayan Guo, Xin Li, Yibing Song, Hao Luo, Fan Wang, et al. Worldvla: Towards autoregressive action world model. *arXiv preprint arXiv:2506.21539*, 2025.

[12] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International conference on machine learning*, pages 794–803. PMLR, 2018.

[13] Yilun Du, Sherry Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Josh Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *Advances in neural information processing systems*, 36:9156–9172, 2023.

[14] Joey Hejna, Chethan Bhateja, Yichen Jiang, Karl Pertsch, and Dorsa Sadigh. Remix: Optimizing data mixtures for large scale imitation learning. *arXiv preprint arXiv:2408.14037*, 2024.

[15] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[16] Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U Tan, Navonil Majumder, Soujanya Poria, et al. Nora: A small open-sourced generalist vision language action model for embodied tasks. *arXiv preprint arXiv:2504.19854*, 2025.

[17] Guanhua Ji, Harsha Polavaram, Lawrence Yunliang Chen, Sandeep Bajamahal, Zehan Ma, Simeon Adebola, Chenfeng Xu, and Ken Goldberg. Oxe-auge: A large-scale robot augmentation of oxe for scaling cross-embodiment policy learning. *arXiv preprint arXiv:2512.13100*, 2025.

[18] Ran Jiang et al. Lda-1b: Scaling latent dynamics action model via universal embodied data ingestion. *arXiv preprint arXiv:2602.12215*, 2026.

[19] JoyAI-RA Team. Joyai-ra 0.1: A foundation model for robotic autonomy. *arXiv preprint arXiv:2604.20100*, 2026.

[20] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, et al. Droid: A large-scale in-the-wild robot manipulation dataset. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

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

[22] Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

[23] Moo Jin Kim, Yihuai Gao, Tsung-Yi Lin, Yen-Chen Lin, Yunhao Ge, et al. Cosmos policy: Fine-tuning video models for visuomotor control and planning. *arXiv preprint arXiv:2601.16163*, 2026.

[24] Haizhou Li et al. Lap: Language-action pre-training enables zero-shot cross-embodiment transfer. *arXiv preprint arXiv:2602.10556*, 2026.

[25] Sylvia Li et al. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

[26] Xuanlin Li et al. Evaluating real-world robot manipulation policies in simulation. *arXiv preprint arXiv:2405.05941*, 2024.

[27] 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. *arXiv preprint arXiv:2508.20072*, 2025.

[28] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

[29] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

[30] Hao Luo, Wanpeng Zhang, Yicheng Feng, Sipeng Zheng, et al. Being-h0.7: A latent world-action model from egocentric videos. *arXiv preprint arXiv:2605.00078*, 2026.

[31] Shangchen Miao, Ningya Feng, Jialong Wu, Ye Lin, Xu He, Dong Li, and Mingsheng Long. Jepa-vla: Video predictive embedding is needed for vla models. *arXiv preprint arXiv:2602.11832*, 2026.

[32] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration 0. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[33] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

[34] 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. *arXiv preprint arXiv:2501.09747*, 2025.

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

[36] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: international conference for high performance computing, networking, storage and analysis*, pages 1–16. IEEE, 2020.

[37] Hao Shi, Bin Xie, Yingfei Liu, Lin Sun, Fengrong Liu, Tiancai Wang, Erjin Zhou, Haoqiang Fan, Xiangyu Zhang, and Gao Huang. Memoryvla: Perceptual-cognitive memory in vision-language-action models for robotic manipulation. *arXiv preprint arXiv:2508.19236*, 2025.

[38] Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, et al. Vla-jepa: Enhancing vision-language-action model with latent world model. *arXiv preprint arXiv:2602.10098*, 2026.

[39] Shuhan Tan, Kairan Dou, Yue Zhao, and Philipp Krähenbühl. Interactive post-training for vision-language-action models. *arXiv preprint arXiv:2505.17016*, 2025.

[40] 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. *arXiv preprint arXiv:2405.12213*, 2024.

[41] Wenhui Wang et al. World action models are zero-shot policies. *arXiv preprint arXiv:2602.15922*, 2026.

[42] Youpeng Wen, Junfan Lin, Yi Zhu, Jianhua Han, Hang Xu, Shen Zhao, and Xiaodan Liang. Vidman: Exploiting implicit dynamics from video diffusion model for effective robot manipulation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[43] Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, et al. Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv preprint arXiv:2412.13877*, 2024.

[44] Shihan Wu, Xuecheng Liu, Shaoxuan Xie, Pengwei Wang, et al. Robocoin: An open-sourced bimanual robotic data collection for integrated manipulation. *arXiv preprint arXiv:2511.17441*, 2025.

[45] Wei Wu, Fan Lu, Yunnan Wang, Shuai Yang, Shi Liu, et al. A pragmatic vla foundation model. *arXiv preprint arXiv:2601.18692*, 2026.

[46] Yandan Yang, Shuang Zeng, Tong Lin, Xinyuan Chang, Dekang Qi, et al. Abot-m0: Vla foundation model for robotic manipulation with action manifold learning. *arXiv preprint arXiv:2602.11236*, 2026.

[47] Yifan Yang, Zhixiang Duan, Tianshi Xie, Fuyu Cao, Pinxi Shen, Peili Song, Chenyang Zhao, Piaopiao Jin, Guokang Sun, Shaoqing Xu, et al. Fpc-vla: A vision-language-action framework with a supervisor for failure prediction and correction. *Expert Systems with Applications*, page 131742, 2026.

[48] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

A Implementation Details

This appendix supplements the implementation details omitted from Section 3.2. The main text retains only details directly related to the paradigm comparison, while this section supports reproducibility.

A.1 VLM and Action Expert Configuration

All models use Qwen3-VL-4B-Instruct as the vision-language backbone [3]. Its language backbone is a Transformer with $L = 36$ layers, hidden dimension $d_{\text{vlm}} = 2048$, and $H = 16$ attention heads. It uses grouped-query attention (GQA) with $H_{\text{kv}} = 8$ key-value heads and head dimension $d_h = 128$. The action expert consists of $N = 36$ DiT blocks with internal hidden dimension $d_a = 1280$. To support concatenation with VLM caches, the key-value head count and head dimension of the action expert are aligned with those of the VLM.

The action expert input is a sequence of noised action tokens with default length $T = 16$. The framework retains interfaces for robot-state tokens and learnable context tokens, but they are not enabled in the experiments reported here. MindWPI and MindLWPI additionally concatenate current-frame latent-space tokens as a prefix to the action expert.

A.2 Key-Value Cache Sharing

During the VLM forward pass, we store the key-value cache at every layer, $\{(\mathbf{K}_l^{\text{vlm}}, \mathbf{V}_l^{\text{vlm}})\}_{l=1}^{L}$, where

$$\mathbf{K}_l^{\text{vlm}}, \mathbf{V}_l^{\text{vlm}} \in \mathbb{R}^{B \times H_{\text{kv}} \times S_{\text{vlm}} \times d_h}. \tag{8}$$

At action-expert layer i, queries, keys, and values $\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i$ are produced from the action-expert tokens. The VLM cache at layer i is concatenated with the action expert's own keys and values:

$$\tilde{\mathbf{K}}_i = [\mathbf{K}_i^{\text{vlm}}; \mathbf{K}_i], \qquad \tilde{\mathbf{V}}_i = [\mathbf{V}_i^{\text{vlm}}; \mathbf{V}_i]. \tag{9}$$

The attention at this layer is

$$\text{Attn}(\mathbf{Q}_i, \tilde{\mathbf{K}}_i, \tilde{\mathbf{V}}_i) = \text{softmax}\left(\frac{\mathbf{Q}_i \tilde{\mathbf{K}}_i^{\top}}{\sqrt{d_h}} + \mathbf{M}\right) \tilde{\mathbf{V}}_i. \tag{10}$$

Action-expert layers and VLM layers are aligned layer by layer by default, although a layer-offset parameter can also be used. At inference time, the VLM cache and static prefix cache can be reused across denoising steps to reduce control latency.

A.3 Attention Mask

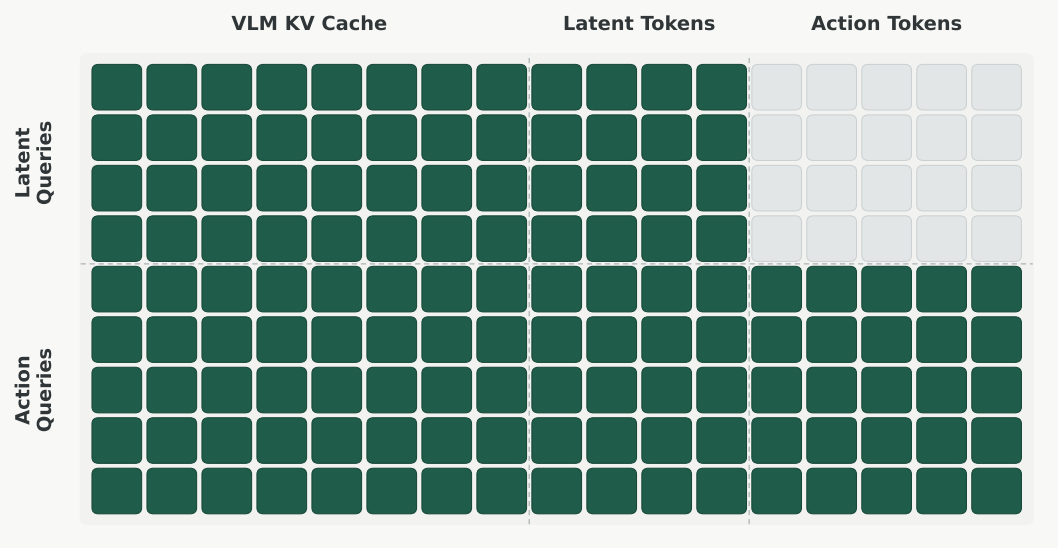

Figure 5: Attention mask used in MindWPI and MindLWPI. Rows denote query tokens and columns denote key/value tokens. Latent queries can attend to the VLM KV cache and latent tokens, but are masked from action tokens to prevent shortcut prediction. Action queries can attend to the VLM KV cache, latent tokens, and action tokens, allowing action generation to use predictive latent context.

All action-expert tokens can attend to the VLM cache. In MindPI and MindLPI, the action-expert sequence contains only action tokens, and action tokens are mutually visible. In MindWPI and MindLWPI, the action-expert sequence contains latent tokens and action tokens. To prevent future latent prediction from exploiting the action trajectory as a shortcut, latent tokens cannot attend to action tokens; action tokens can attend to both latent tokens and other action tokens. In other words, latent prediction can rely only on current vision-language context and current latent representations, while action generation can use predictive latent context.

DiT blocks inject the flow-matching timestep condition using AdaLN: the timestep is encoded with sinusoidal positional encoding and an MLP into shift, scale, and gate parameters for attention and feed-forward layers. Action tokens use RoPE positions and are appended after the VLM input sequence.

B MindLPI Language Supervision Details

MindLPI and MindLWPI use two formats for action verbalization.

Discrete integer format. Action values normalized to $[-1, 1]$ are uniformly discretized into 1000 bins and represented as integer sequences. This format does not depend on physical units and is suitable for data with different action spaces or without a unified physical scale.

Natural-language format. We first extract the 7-dimensional right-arm action and recover the normalized action to physical units according to the global quantiles (q_{01}, q_{99}):

$$\mathbf{a}^{\text{phys}} = \frac{1}{2}(\mathbf{a}^{\text{norm}} + \mathbf{1}) \odot (q_{99} - q_{01}) + q_{01}. \tag{11}$$

We then sum over the 16-step action chunk, convert translation to centimeters, convert rotation to degrees and round it to 5°, and determine gripper open/close state according to a threshold. The final text has the form "move forward 12 cm, move up 8 cm, close gripper." When downstream actions are joint-space increments or lack physical units, we use the discrete integer format.

The current language supervision is provided only by LAP-style action-description templates. MindLPI optimizes $\mathcal{L}_{\text{act}} + \lambda_{\text{lang}}\mathcal{L}_{\text{lang}}$ during pre-training, where $\lambda_{\text{lang}} = 0.1$. MindLWPI additionally adds the future latent loss and uses the same language loss weight.

C MindWPI and MindLWPI Latent Prediction Details

MindWPI uses a frozen V-JEPA 2 model as the latent feature extractor [2]. Given current and future frames, it extracts $\mathbf{z}_{\text{cur}}, \mathbf{z}_{\text{fut}} \in \mathbb{R}^{B\times N_z\times d_z}$. During pre-training, future frames are sampled with a fixed offset, whose default value is 8 frames. After freezing the feature extractor, we do not add an external linear projection directly, in order to avoid projecting the features into a low-variance representation; latent encoding and decoding are handled by internal modules of the action expert.

In the currently reported full pre-training experiments for MindWPI/MindLWPI, whenever both action loss and future latent loss are enabled, the pre-training stage uniformly uses $\lambda^{\text{pt}}_{\text{act}} = 1.0$ and $\lambda^{\text{pt}}_{\text{lat}} = 1.0$. The notation $\lambda_{\text{lat}}/\lambda_{\text{act}} = 0.1$ in the main experiments refers only to downstream fine-tuning, namely $\lambda^{\text{ft}}_{\text{lat}} = 0.1$ and $\lambda^{\text{ft}}_{\text{act}} = 1.0$. Downstream fine-tuning may choose whether to retain the latent loss depending on the experimental setting.

AvgPool-k4 compression. MindLWPI compresses the 256 V-JEPA 2 latent tokens with AvgPool-k4 by default: every 4 adjacent tokens along the token sequence are averaged, resulting in 64 compressed tokens. This operation is applied to both the current latent and the future latent target. We also tested MLP compression, where an MLP first compresses the hidden dimension to $1/k$ and then concatenates every k tokens into one token. However, experiments show that simple average pooling is already sufficiently effective, so AvgPool-k4 is used by default in the main text.

D Training and Fine-tuning Hyperparameters

D.1 Pre-training Data and Sampling

The OXEMix pre-training data mixture consists of DROID, raw OpenX-Embodiment data, OpenX-Augmented embodiment-augmented data, and the RoboCOIN subset, totaling approximately $5{,}000$ hours [20, 32, 44]. In the text, OXE raw subset denotes the raw OXE subset including DROID but excluding OXE-Augmented and RoboCOIN. All data are converted into the LeRobot format and mapped into the 14-dimensional action space. To alleviate oversampling caused by large differences in dataset size, the sampling probability considers both dataset scale and trajectory length. MindWPI and MindLWPI use world-model variants of the same data mixture and additionally provide current and future frames. MindLPI and MindLWPI additionally use action-description text as language supervision.

D.2 Optimization Settings

All four paradigms use the AdamW optimizer with $\beta = (0.9, 0.95)$. The learning rate is 1×10^{-5} for the VLM backbone and 1×10^{-4} for the action expert. The learning-rate schedule is cosine decay with a minimum learning rate of 5×10^{-7} and a warmup of $5{,}000$ steps. Training uses bfloat16 mixed precision, gradient checkpointing, and gradient clipping with a maximum gradient norm of 1.0. The maximum number of pre-training steps is 2×10^{5}. Distributed training uses data parallelism and ZeRO-2 optimization [36]. During pre-training, neither the VLM nor the action expert is frozen, except in the MindPI (Frozen VLM) control experiment.

The flow-matching timestep is sampled as $u \sim \mathrm{Beta}(1.5, 1.0)$ and transformed by $t = (s_0 - u)/s_0$, where $s_0 = 0.999$. To improve timestep coverage, each sample is replicated 4 times within a batch, and each replica independently samples (t, ϵ). At inference time, we use 4 Euler integration steps.

D.3 Downstream Full-Parameter Fine-tuning

Downstream fine-tuning is conducted on LIBERO, LIBERO-Plus, and SimplerEnv. The model architecture remains consistent with pre-training so that checkpoints can be loaded directly. Taking LIBERO as an example, the native Franka single-arm 7-dimensional joint-space increment is zero-padded into the 14-dimensional action space, and a validity mask marks the true action slots. Fine-tuning is full-parameter by default, uses a maximum of 1×10^{5} steps, and has a warmup of $2{,}000$ steps. Other optimization settings are the same as in pre-training.

E Low-Rank Adaptation Settings

Considering the storage and computational cost of full-parameter fine-tuning, we additionally evaluate low-rank adaptation with LoRA [15]. We inject low-rank modules only into the query and value projection matrices of attention layers and compare three injection strategies.

Action-side LoRA. We freeze the VLM and inject LoRA only into the query and value projections of attention layers in the action expert.

VLM-side LoRA. The action expert is fully trainable, and LoRA is injected only into the query and value projections of attention layers in the VLM language backbone. This configuration introduces approximately 94.37M trainable parameters when $r = 256$.

Both-side LoRA. LoRA is injected into both the VLM and action expert to evaluate adaptation gains under a larger parameter budget.

These settings compare whether the model can retain the transfer advantages of pre-training under different trainable-parameter budgets. The LoRA results are reported as an efficient-adaptation experiment and are not a primary variable in the fair comparison of the four training paradigms. Table 12 reports complete results for the four LIBERO suites.