Endowing Your Vision-Language-Action Model with a Predictive Mind

Pengxiang Ding[1,2,*] **Haoying Wang**[3,*] **Minghui Lin**[1] **Qishen Wang**[4]
Zhenyu Ding[3] **Runze Suo**[5] **Xuanxuan An**[3] **Wenxuan Song**[6]
Fuhao Li[6] **Han Zhao**[1,2] **Donglin Wang**[1,†] **Ning Ding**[3,†]

[1]Westlake University [2]Zhejiang University
[3]State Key Laboratory of Human-Machine Hybrid Augmented Intelligence,
Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University
[4]University of Science and Technology of China
[5]Fudan University
[6]The Hong Kong University of Science and Technology (Guangzhou)
[*]Equal contribution. [†]Corresponding authors.
Donglin Wang: wangdonglin@westlake.edu.cn
Ning Ding: ding.ning@xjtu.edu.cn

Abstract

Vision-Language-Action (VLA) models transfer semantic priors from large-scale vision-language models to robotic manipulation, but their backbones are typically optimized for static visual understanding rather than action-conditioned dynamics. Consequently, the features used by the policy may lack predictive information about how the scene evolves under the robot's actions. Existing future-prediction methods either reconstruct pixels, overemphasizing appearance details irrelevant to control, or use auxiliary latent predictors that remain weakly coupled with the policy-consumed backbone features. We propose **PredMind**, a plug-and-play predictive representation learning framework that treats future prediction as a training signal for the VLA backbone itself. Instead of reconstructing future pixels, PredMind aligns intermediate backbone representations with their future counterparts conditioned on executed actions, shifting supervision from low-level appearance to the model's semantic feature space. This directly injects action-conditioned temporal structure into the representation hierarchy used for control while preserving pretrained VLM priors. All auxiliary prediction components are discarded after training, introducing no architectural changes, additional parameters, or inference latency at deployment. Across simulation benchmarks, real-world manipulation tasks, and multiple VLA backbones, **PredMind** consistently improves learning efficiency and final performance. Especially on LIBERO, it matches or surpasses a fully trained base VLA using only **1/60** of the training iterations, demonstrating the benefit of embedding predictive dynamics directly into backbone features used for control.

1 Introduction

Vision-Language-Action (VLA) models have emerged as a promising paradigm for general-purpose robotic manipulation by transferring semantic and commonsense priors from large-scale Vision-Language Models (VLMs) into physical action spaces [19, 10, 2, 27, 14]. However, most VLA models inherit backbones that are primarily optimized for static visual-language understanding. As a result, although these backbones provide strong object- and scene-level semantics, they do not explicitly encode how the scene will evolve under the robot's own actions. This raises a key challenge: how to incorporate future visual dynamics into VLA models to better support action decision-making.

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

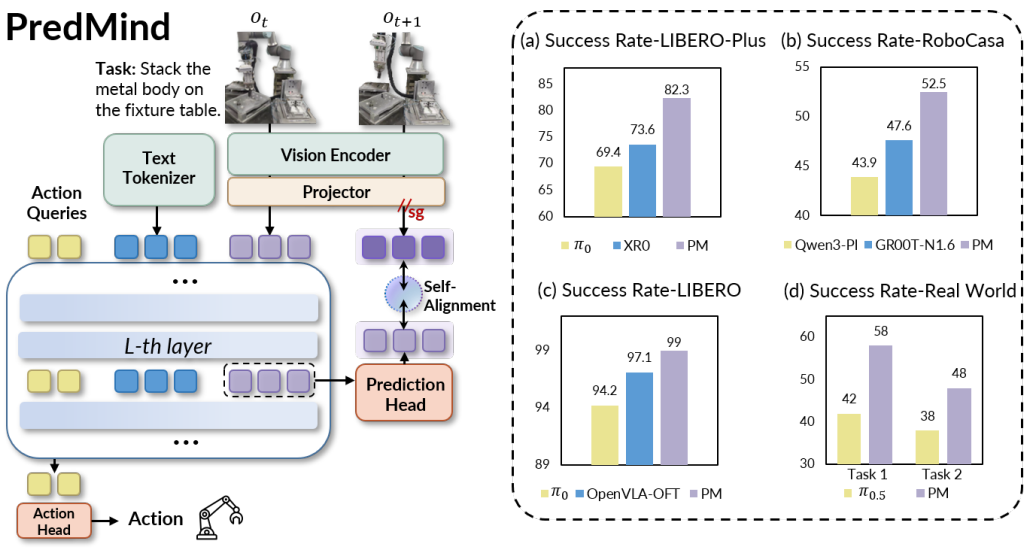

Figure 1: **Left**: Illustration of **PredMind** using OpenVLA-OFT [11] as an example backbone. **PredMind** promotes the internalization of dynamic prediction through self-representation alignment in the VLA latent space. "sg" denotes stop-gradient. **Right**: (a–d) We demonstrate consistent performance gains across diverse evaluation settings, including LIBERO-Plus [6], RoboCasa [17], LIBERO [15], and real-world robotic experiments.

Existing future-prediction strategies only partially address this issue. Pixel-level world models supervise future frames or subgoals [4, 34], but such objectives are overly detailed for policy learning: they penalize changes in lighting, texture, background, and other visual factors that are often irrelevant to manipulation. More importantly, pixel reconstruction encourages the model to spend capacity on appearance-level fidelity rather than on the semantic and geometric changes that determine the next action. Latent prediction methods reduce this burden by predicting compact future tokens or auxiliary embeddings [35, 39]. Yet in many cases, the learned predictive representation is introduced through an additional module, while the original backbone features that drive the action head remain only weakly modified. Thus, the policy may benefit from an auxiliary training signal, but the core representation hierarchy is not fundamentally made predictive.

We further examine where temporal and predictive information resides in existing VLA architectures. Although modern VLA models inherit strong static semantic priors from VLMs, these representations are not naturally optimized to encode action-conditioned future dynamics. As a result, temporal reasoning may still rely heavily on downstream policy heads learning from limited robot data. This suggests that the bottleneck is not merely the absence of an auxiliary dynamics module, but the lack of predictive structure in the representations that the policy actually consumes. This observation motivates us to ask whether future prediction should be treated not as an external add-on, but as a training signal for the backbone itself. Based on these observations, we propose a predictive representation learning framework for VLA models, where future prediction is used to directly shape the representation hierarchy itself. Instead of generating future pixels, we align intermediate representations with their future counterparts under the executed action, shifting predictive supervision from low-level appearance reconstruction to the model's own semantic feature space. By applying supervision directly to intermediate layers, the learned dynamics become integrated into the same representations used for action generation, rather than being isolated in a separate prediction module.

We instantiate this idea as **PredMind**, a plug-and-play training framework for VLA models. Given robot trajectories, **PredMind** constructs self-supervised future-aware objectives over backbone features: current representations are encouraged to encode information predictive of future representations conditioned on the robot's actions. The supervision is applied inside the backbone, rather than only at the policy output, so that predictive structure is learned before action decoding. This design has two important properties. First, it preserves the semantic prior of the pretrained VLM backbone while augmenting it with action-conditioned temporal structure. Second, it introduces no

deployment-time cost: all auxiliary prediction components are used only during training, and the final policy keeps the same architecture, parameter count, and inference latency as the original model.

Empirically, this representation-level intervention leads to consistent gains. Across simulation benchmarks, real-robot manipulation tasks, and multiple VLA backbones, **PredMind** improves both learning efficiency and final performance. In particular, on LIBERO benchmarks, it matches or surpasses a fully trained base VLA using only **1/60** of the training iterations (see Appendix A), indicating that predictive backbone features substantially reduce the amount of robot data optimization required. **PredMind** also continues to improve performance at convergence, suggesting that the benefit is not merely faster optimization but a better action-relevant representation. These results support our central claim: for VLA models, predictive dynamics are most effective when they are embedded into the backbone features used for control, rather than supplied through external future prediction modules.

2 Related Work

Recent work augments VLA with explicit future prediction. Broadly, these methods differ by the prediction space: forecasting future observations in pixel space, or in a latent representation space.

Prediction in Pixel Space. Pixel-space approaches supervise dynamics by forecasting future images (often conditioned on candidate actions) alongside policy learning. Representative designs include generating subgoal/future frames as intermediate visual plans and conditioning action decoding on these predictions [37, 34, 4, 30]. A related paradigm predicts future observations first and then learns actions via inverse dynamics between the current and predicted frames [28, 16]. DUST [31] further separates action and future-frame generation into dual diffusion streams with cross-modal exchange, mitigating interference from forcing both modalities into a shared generation space. A common limitation is that pixel-space forecasting is expensive and appearance-sensitive, so the learned dynamics may overfit to visual details rather than capturing task-relevant state transitions.

Prediction in Latent Representation Space. To avoid pixel-level reconstruction overhead and improve robustness, another line predicts futures in a latent space, often distilling supervision from pre-trained vision foundation models (VFMs) [21, 18, 22, 12, 32, 33]. DreamVLA [35] uses VFM features as semantic lookahead targets; FRAPPE [36] predicts future latents and further aligns to multiple VFM spaces in post-training for better generalization; FLARE [39] injects a small set of learnable future tokens and aligns them to future latents to encourage implicit world modeling. However, because supervision is anchored to external VFM representations, the dynamics signal is often treated as auxiliary guidance rather than being internalized into the VLM's semantic space.

3 Preliminaries

3.1 Problem Formulation

Definition. We formulate language- and vision-conditioned robotic manipulation as a Markov Decision Process. At each step t, the agent receives an observation $o_t \in \mathbb{R}^{H\times W\times C}$ and a language instruction l, and predicts an action $a_t \in \mathcal{A}$ to accomplish the task. A VLA policy π_θ typically consists of a vision-language backbone Φ and an action head π_{head}. The observation o_t and instruction l are encoded into a shared latent space by Φ, and the resulting representation is decoded into an action:

$$\hat{\mathbf{a}}_t = \pi_\theta(\mathbf{o}_t, \boldsymbol{l}) = \pi_{head}(\Phi(o_t, \boldsymbol{l})). \tag{1}$$

Given a dataset of expert trajectories $\mathcal{D} = \{\tau_i\}_{i=1}^N$ with $\tau = \{(\mathbf{o}_t, \boldsymbol{l}, \mathbf{a}_t)\}_{t=1}^T$, the policy is trained via behavioral imitation by minimizing the discrepancy between predicted and expert actions:

$$\mathcal{L}_A = \mathbb{E}_{\tau\sim\mathcal{D},\, t}\left[\mathcal{L}_{act}(\hat{\mathbf{a}}_t, \mathbf{a}_t)\right], \tag{2}$$

where $\mathcal{L}_{act}$ denotes the task-specific action loss.

Limitation. While effective for reactive control, this formulation does not explicitly model how the scene evolves under robot actions. This limitation has motivated recent VLA models to incorporate dynamics information through world-modeling objectives.

3.2 Analysis on Existing World-modeling Objectives

VLA policies inherit strong semantic priors from pre-trained VLM backbones, yet these backbones are primarily optimized for static visual-language understanding rather than action-conditioned dynamics. Consequently, temporal reasoning is often delegated to the downstream action head and must be recovered from limited robot demonstrations. We conduct a diagnostic study to understand *when* future-aware training improves VLA control and *where* the predictive signal should be injected.

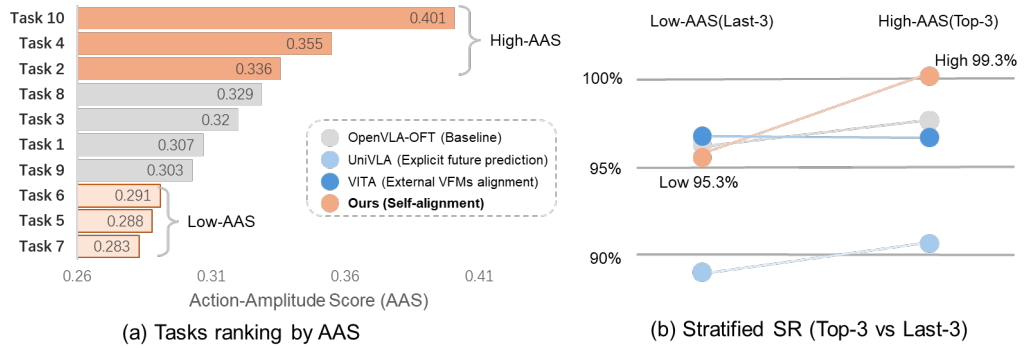

Figure 2: **Control-intensity stratification.** (a) Tasks ranked by action-amplitude score (AAS); Top-3 and Last-3 highlighted. (b) Success rates (SR) on Low-AAS (Last-3) and High-AAS (Top-3) tasks.

Task Stratification. We use action magnitude as a simple proxy for how strongly a task depends on action-driven state transitions. For each task $\mathcal{T}$, we define the **Action-Amplitude Score (AAS)** from expert demonstrations:

$$\mathrm{AAS}(\mathcal{T}) = \frac{1}{d_a} \sum_{d=1}^{d_a} \mathbb{E}_{\tau \sim \mathcal{D}_{\mathcal{T}},\, t} \left[\left| a_t^{(d)} \right| \right], \tag{3}$$

where d_a is the action dimensionality and $a_t^{(d)}$ denotes the d-th action component at time t. Higher AAS corresponds to stronger actuation and larger action-induced state changes. We rank tasks by AAS and split each benchmark into **High-AAS** and **Low-AAS** subsets using the top-k and bottom-k tasks, with k=3. Fig. 2(a) shows the split on LIBERO-10.

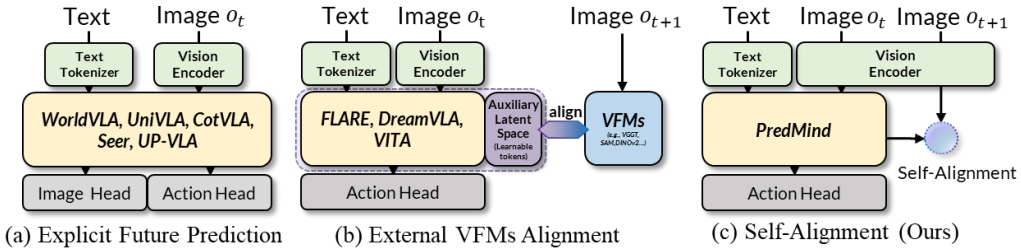

Figure 3: **Three ways to inject dynamics into VLA training.**

Future-injection Strategies. We compare three ways of incorporating future information into VLA training (Fig. 3): *explicit future prediction*, which predicts future visual states for action decoding [30]; *external predictive alignment*, which distills dynamics into an auxiliary latent space [5]; and *native backbone self-alignment*, which aligns the current representation of o_t with the future representation of o_{t+1} extracted by the same backbone. The last variant directly supervises the features consumed by the policy head and serves as a diagnostic pilot for our final design.

Design Principle. As shown in Fig. 2(b), future-aware supervision yields modest gains on Low-AAS tasks, where reactive visual control is often sufficient, but substantially larger gains on High-AAS tasks that require anticipating action-induced state changes. Among the compared strategies, native backbone self-alignment performs best. This indicates that the benefit of future-aware training depends less on adding an auxiliary prediction target and more on whether predictive gradients

reshape the VLA backbone features used for action generation. This finding motivates the design of our architecture, which injects action-conditioned predictive supervision directly into intermediate VLA backbone representations during training, while leaving the deployment-time policy architecture and inference latency unchanged.

4 PredMind

4.1 Overall Framework

To endow VLA models with intrinsic dynamic modeling capability, we introduce a predictive alignment objective at intermediate layers of the VLM backbone. Given two consecutive observations o_t and o_{t+1}, we enforce the intermediate representation of the current observation to align with a future representation target. Crucially, the alignment is imposed directly on the native visual tokens of the VLM, without introducing auxiliary latent tokens. We introduce a predictive alignment objective at an intermediate layer k of the VLM backbone. Let $z_t^k = \Phi^k(o_t, l)$ denote the intermediate semantic representation of the current observation at layer k. We employ a lightweight prediction head MLP $g_\psi(\cdot)$ to produce a future representation estimate: $\hat{z}_{t+1}^k = g_\psi(z_t^k)$. The target future representation z_{t+1}^{target} is obtained either from a frozen expert encoder or from the native VLA backbone itself, depending on the alignment strategy.

In OpenVLA-OFT, bidirectional attention allows the visual-token representations to incorporate information from the language instruction. The predictor therefore operates on instruction-conditioned features $z_t^k = \Phi^k(o_t, l)$, providing language-goal conditioning through the backbone representation.

The dynamic alignment loss is defined as:

$$\mathcal{L}_{\text{pm}} = \mathbb{E}_{\tau \sim \mathcal{D},\, t} \left[\mathcal{S}\left(g_\psi(z_t^k), z_{t+1}^{\text{target}} \right) \right], \tag{4}$$

where $\mathcal{S}(\cdot, \cdot)$ denotes a distance metric (MSE loss and cosine distance).

The overall training objective combines action imitation and predictive alignment:

$$\mathcal{L} = \mathcal{L}_{\text{A}} + \lambda \mathcal{L}_{\text{pm}}, \tag{5}$$

where λ controls the trade-off between behavioral cloning and intrinsic dynamic regularization.

4.2 Model Recipes

In this section, we use the principles from the previous section to design a number of model recipes. Our objective is to facilitate the VLA's ability to internalize future state transitions directly within the latent representation space. As shown in Figure 4, we instantiate the above principles along three orthogonal design axes: (1) the feature space where alignment is performed, (2) the source of future representations, and (3) the depth at which alignment is applied.

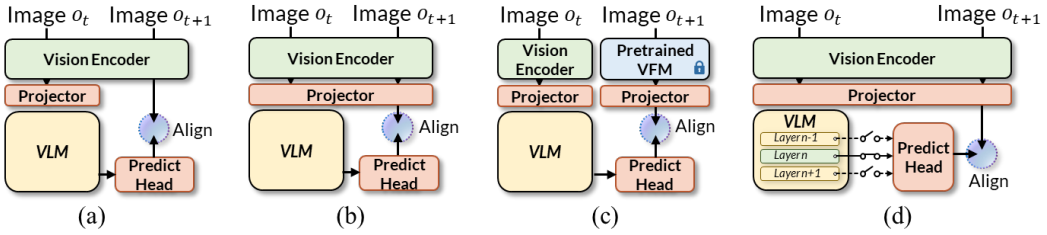

Figure 4: **Model Recipe Variants.** (a) Self-representation alignment in the visual feature space (vision encoder). (b) Self-representation alignment in the VLM semantic space (VLM latent). (c) Alignment with external expert representations (VFMs). (d) Ablation on intermediate-layer alignment.

Visual Feature Space Alignment or VLM Semantic Space Alignment. We align the current visual representation with the future visual representation in latent space. A critical question, however, is in which feature space the alignment should be performed. We consider two alternative latent spaces: the visual feature space and the VLM semantic space. In the visual feature alignment scheme, the intermediate representation of z_t^k is projected via a prediction head into the visual feature space.

This projected embedding is then aligned with the latent representation of z_{t+1}^{target} obtained from a pretrained expert encoder. In contrast, VLM semantic alignment operates entirely within the intrinsic semantic space of the VLM model. Both o_t and o_{t+1} are processed through the shared visual encoder and projection layers to produce embeddings in the VLM latent space.

External Expert Representation Alignment or Self-Representation Alignment. Guided by the above design principle, we directly impose future alignment constraints on the native visual tokens of the VLM, eliminating the need to introduce auxiliary latent tokens. The first strategy aligns the predicted future representation with features extracted from external visual foundation models (VFMs). In this setting, the future observation o_{t+1} is encoded by a pretrained expert encoder that provides expert representations. The current observation o_t VLA latent tokens is projected into this expert space through a lightweight prediction head, and the model is trained to minimize the discrepancy between the two. Alternatively, for self-representation alignment, both o_t and o_{t+1} are fed into a shared vision encoder to produce current and future embeddings. The current observation embedding propagates through the VLM backbone to an intermediate layer, where it is mapped by a prediction head to align directly with the future visual embedding. Unlike expert alignment, this variant does not introduce external representational anchors.

Layer-wise Dynamic Alignment. Dynamic information may manifest differently across representation depths. Early layers emphasize local geometric features, higher layers capture abstract semantics, while intermediate layers integrate structural and semantic information. We empirically study how alignment at different depths affects intrinsic dynamic encoding and downstream policy performance.

5 Experiments

Our experimental goal is to systematically evaluate the proposed training variants and understand their impact on downstream task performance and representation structure. Rather than solely reporting policy improvements, we aim to analyze how different alignment strategies influence the internalization of environment dynamics within VLA models. Specifically, we seek to answer the following questions:

1. *Which alignment design choice is most effective, and what are the underlying factors?*
2. *How does **PredMind** compare with existing VLA paradigms on manipulation tasks?*
3. *How does **PredMind** perform on real-world manipulation tasks?*

5.1 Model Variants Selection

Experiments Setup. We conduct controlled experiments on the LIBERO benchmark to evaluate the model variants proposed in Section 4.2. These experiments aim to isolate the effects of alignment space, supervision source, and layer selection, and to determine which design choices most effectively promote intrinsic dynamic modeling within the VLA framework.

Results. Table 1 presents ablation results for different design choices (see Section 4.2). Aligning in the visual feature space leads to a noticeable performance drop compared with alignment in the native VLM latent space, decreasing the average success rate by 4.7%. This suggests that alignment within the VLM latent space better facilitates internalizing future dynamics. We further compare different external experts. SigLIP performs best, likely due to its image–text pretraining that provides stronger semantic representations suited for multimodal VLA models, while the DINO series mainly captures visual invariances through self-supervised learning. Nevertheless, all external experts remain inferior to **PredMind** self-representation alignment, highlighting the advantage of operating within the model's native representation space. Finally, aligning the final layer consistently achieves the best performance. Notably, our method trained for only 2 epochs on a single GPU surpasses the baseline trained for 150K iterations on 8 GPUs (over 95 epochs), demonstrating much faster convergence.

Why does alignment in the VLM space outperform alignment in a visual feature space? When alignment is enforced in the visual feature space, the model learns to project its intermediate features into a separate geometry defined by an external encoder. In this setting, the dynamic signal can be largely accommodated by the prediction head or projection mapping, rather than genuinely reshaping the intrinsic latent representations of the VLM itself. In contrast, aligning directly within the VLM

Table 1: Ablation study over different design choices, including the alignment space, target representation, alignment layer within the VLA backbone, and the number of training iterations.

Alignment Space	Target Representation	Aligned Layerth	Training Iterations	Spatial SR (%)	Object SR (%)	Goal SR (%)	Long SR (%)	Average SR (%)
×	×	×	150K(8GPU)	97.6	98.4	97.9	94.5	97.1
Visual Feature	Self	32	2epochs	97.4	98.6	96.6	84.6	94.3
VLM Latent	Self	32	2epochs	**99.4**	**100**	**99.2**	**97.4**	**99.0**
VLM Latent	SigLIP	32	2epochs	99.0	99.8	98.2	92.6	97.4
VLM Latent	DINOv2	32	2epochs	97.6	97.0	97.8	93.0	96.4
VLM Latent	DINOv3	32	2epochs	98.8	97.0	97.0	94.0	96.7
VLM Latent	Self	1	2epochs	98.0	99.6	96.8	92.0	96.6
VLM Latent	Self	8	2epochs	96.4	99.6	98.2	93.0	96.8
VLM Latent	Self	16	2epochs	97.4	99.0	95.8	95.0	96.8
VLM Latent	Self	24	2epochs	97.8	99.2	98.2	94.8	97.5
VLM Latent	Self	32	2epochs	**99.4**	**100**	**99.2**	**97.4**	**99.0**
VLM Latent	Self	32	2K	1.6	1.4	16.4	1.8	5.3
VLM Latent	Self	32	5K	76.8	79.4	49.0	39.4	61.2
VLM Latent	Self	32	10K	88.2	93.6	69.4	71.6	80.7
VLM Latent	Self	32	20K	91.6	99.0	85.4	95.2	92.8
VLM Latent	Self	32	2epochs	**99.4**	**100**	**99.2**	**97.4**	**99.0**

latent space ensures that dynamic information is embedded into the model's semantic structure, rather than mediated by cross-space mappings.

Why is external VFMs expert alignment inferior to self-representation alignment? A key reason is representational mismatch. VFMs are typically trained with objectives such as classification or contrastive learning, emphasizing appearance-level invariances that may not align with control-relevant state transitions. Aligning to such external features can introduce biases suboptimal for policy learning. In contrast, self-representation alignment operates within a shared encoding space, where both o_t and o_{t+1} are processed by the same encoder. This encourages the model to internalize dynamics as transformations within its own semantic space, rather than approximating externally defined features. Overall, effective dynamics modeling relies more on structuring intrinsic representations than on stronger external supervision.

Which intermediate layer is most effective for alignment? We find that aligning at the final semantic layer yields the best performance. Early layers capture local structures and low-level geometry, where temporal alignment mainly enforces short-term visual consistency without directly affecting decision-relevant representations. In contrast, late-layer features encode high-level semantics closely coupled with the policy head. Aligning at this level directly shapes action-relevant representations, embedding temporal structure into control features. Moreover, higher-level latent representations are more compact and semantically organized, which may facilitate the internalization of temporal evolution within the representation space. This likely explains why late-layer alignment consistently produces greater improvements than constraints applied at lower levels.

Training Efficiency Analysis. We evaluate whether **PredMind** improves training efficiency on LIBERO. Results in Table 1 show the relationship between success rate and training iterations for the baseline and our method. **PredMind** significantly accelerates convergence: on LIBERO-Long, the baseline requires 150k steps to reach a performance that our method achieves with only 20k steps on a single GPU ($\approx$ 2.5k iterations normalized to 8 GPUs), yielding nearly a 60× lower GPU-step budget. We attribute this acceleration to the predictive semantic alignment introduced during training. By providing future-state supervision directly in the VLM semantic space, **PredMind** guides representation learning toward temporally coherent dynamics, reducing the optimization burden of reactive policy learning.

Additional Analysis. We further examine language conditioning and predictive representations in Appendix C. On XR-0, allowing visual tokens to attend to language instructions improves performance in six of the seven LIBERO-Plus perturbation settings compared with the variant without such access. We also report the evolution of the predictive alignment loss during training.

5.2 Comparison with SOTA

Experimental Setup. We evaluate **PredMind** on three simulated embodied manipulation benchmarks: LIBERO [15], LIBERO-Plus [6], and RoboCasa [17]. LIBERO focuses on compositional generalization across task variants and object configurations, while LIBERO-Plus extends it with seven robustness settings, including changes in lighting, camera viewpoints, object appearances, and scene layouts. RoboCasa features longer-horizon household manipulation tasks with stronger temporal dependencies and more complex object interactions. For each benchmark, we instantiate **PredMind** on top of a strong VLA model representative of current performance in the corresponding simulation environment, and use the same model trained without our objective as the baseline. Specifically, we use OpenVLA-OFT [11] and StarVLA [24] on LIBERO, GR00T N1.6 [1] on RoboCasa, and XR0 [3] on LIBERO-Plus. Further implementation and evaluation details are provided in Section A.

Table 2: **Success rates (%) on LIBERO.** We compare prior VLA methods with **PredMind** instantiated on two VLA backbones. Best results are in bold.

Method	Spatial SR (%)	Object SR (%)	Goal SR (%)	Long SR (%)	Average SR (%)
TraceVLA [38] *[ICLR'25]*	84.6	85.2	75.1	54.1	74.8
OpenVLA [10] *[CoRL'24]*	84.7	88.4	79.2	53.7	76.5
SpatialVLA [20] *[RSS'25]*	88.2	89.9	78.6	55.5	78.1
UniVLA [30] *[ICLR'26]*	95.4	98.8	93.6	94.0	95.5
CoT-VLA [37] *[CVPR'25]*	87.5	91.6	87.6	69.0	83.9
π_0-FAST [19] *[RSS'25]*	96.4	96.8	88.6	60.2	85.5
π_0 [2] *[RSS'25]*	96.8	98.8	95.8	85.2	94.2
$\pi_{0.5}$ [9] *[arXiv'25]*	98.8	98.2	98.0	92.4	96.9
Smol-VLA [23] *[arXiv'25]*	93.0	94.0	91.0	77.0	88.8
GR00T-N1 [1] *[arXiv'25]*	94.4	97.6	93.0	90.6	93.9
VLA-OS [7] *[NeurIPS'25]*	87.0	96.5	92.7	66.0	85.6
VLA-Adapter [29] *[AAAI'26]*	97.8	99.2	97.2	95.0	97.3
Spatial Forcing [13] *[ICLR'26]*	99.4	99.6	98.8	96.0	98.5
GeoVLA [25] *[arXiv'25]*	98.4	99.0	96.6	96.6	97.7
Qwen2.5-VL-GR00T [24] *[arXiv'26]*	97.8	98.2	94.6	90.8	95.4
OpenVLA-OFT [11] *[RSS'25]*	97.6	98.4	97.9	94.5	97.1
PredMind (Qwen2.5-VL-GR00T)	95.4	99.8	95.6	96.6	96.9
PredMind (OpenVLA-OFT)	**99.4**	**100.0**	**99.2**	**97.4**	**99.0**

Results on LIBERO. As shown in Table 2, **PredMind** improves two different VLA backbones on LIBERO. On Qwen2.5-VL-GR00T, it improves the average success rate from 95.4% to 96.9%, with the largest gain on LIBERO-Long (90.8% to 96.6%). Although Spatial decreases from 97.8% to 95.4%, Object, Goal, and Long all improve, indicating that the benefit transfers beyond a single backbone and is strongest on long-horizon manipulation. On OpenVLA-OFT, **PredMind** further reaches the best overall result, achieving **99.0%** average success and improving LIBERO-Long by +2.9 points over the baseline. These gains are obtained without changing the inference-time policy architecture or latency.

Generalization on LIBERO-Plus. We evaluate **PredMind** on LIBERO-Plus [6], which perturbs camera viewpoint, robot initial state, language instruction, lighting, background texture, sensor noise, and object layout. Using Xiaomi-Robotics-0 (XR0) [3] as the backbone, **PredMind** improves the average success rate from 73.6% to **82.3%** on the perturbed benchmark, as shown in Table 3. The improvement is most significant for camera viewpoint (+29.5), sensor noise (+9.1), and robot initial state (+8.2), suggesting stronger generalization under visual and state-distribution shifts. Performance decreases only under lighting perturbations, from 98.0% to 94.9%.

Table 3: **Zero-shot transfer success rates (%) on LIBERO-Plus.** Public baselines are reported from LIBERO-Plus [6]. All methods are evaluated under the zero-shot transfer setting on LIBERO-Plus perturbations. XR0 denotes Xiaomi-Robotics-0 [3], with results based on our own development and implementation. Best results are in bold.

Method	Camera	Robot	Language	Light	Background	Noise	Layout	Avg.
WorldVLA [4] *[arXiv'25]*	0.1	27.9	41.6	43.7	17.1	10.9	38.0	25.0
OpenVLA [10] *[CoRL'24]*	0.8	3.5	23.0	8.1	34.8	15.2	28.5	15.6
NORA [8] *[arXiv'25]*	2.2	37.0	65.1	45.7	58.6	12.8	62.1	39.0
UniVLA [30] *[ICLR'26]*	1.8	46.2	69.6	69.0	81.0	21.2	31.9	42.9
π_0-FAST [19] *[RSS'25]*	65.1	21.6	61.0	73.2	73.2	74.4	68.8	61.6
RIPT-VLA [26] *[arXiv'25]*	55.2	31.2	77.6	88.4	91.6	73.5	74.2	68.4
OpenVLA-OFT [11] *[RSS'25]*	56.4	31.9	79.5	88.7	93.3	75.8	74.2	69.6
π_0^* [2] *[RSS'25]*	61.0	40.8	63.5	89.3	84.1	80.1	76.4	69.4
XR0 [3] *[arXiv'26]*	40.0	52.3	84.9	**98.0**	92.6	82.6	78.5	73.6
PredMind (XR0)	**69.4**	**60.5**	**88.1**	94.9	**96.8**	**91.8**	**82.7**	**82.3**

Results on RoboCasa. Table 4 shows the results on RoboCasa-GR1. **PredMind** obtains the highest overall success rate, 52.5%, outperforming the strongest baseline Qwen3-OFT by 3.7 points. On the *PnP * to * Close* group, the improvement is more pronounced: **PredMind** reaches 60.0%, while the best baseline reaches 50.3%. These tasks require the policy to place an object into an articulated container and finish with the target container state. The result indicates that predictive alignment remains useful beyond LIBERO, especially for tasks with more structured state transitions. We report the full task-level breakdown in Section B.

Table 4: **Success rates (%) on RoboCasa-GR1.** All models are jointly trained on the 24 RoboCasa-GR1 tabletop tasks. We report the articulated-container closing group and the overall average in the main paper.

Method	**PnP * to * Close (6 tasks)**	**Overall (24 tasks)**
Qwen3-GR00T [24] *[arXiv'26]*	50.3	47.8
Qwen3-π [24] *[arXiv'26]*	42.3	43.9
Qwen3-OFT [24] *[arXiv'26]*	43.7	48.8
Qwen3-FAST [24] *[arXiv'26]*	35.0	39.0
GR00T-N1.6 [1] *[arXiv'25]*	24.2	47.6
PredMind (GR00T-N1.6)	**60.0**	**52.5**
Gain over GR00T-N1.6	**+35.8**	**+4.9**

5.3 Real-World Experiments

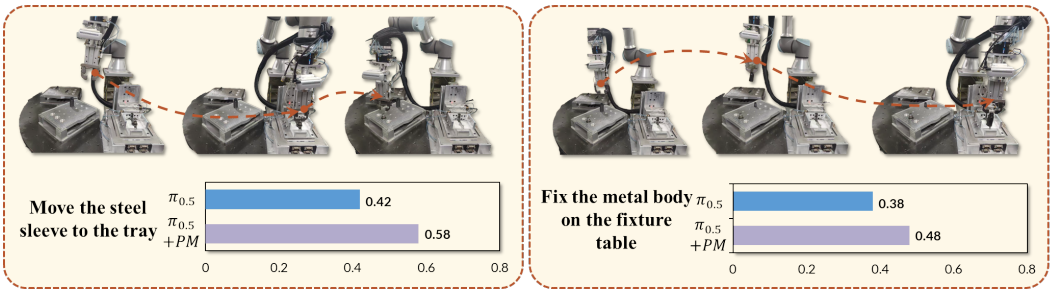

Figure 5: **Real-World Deployment.** We deploy our approach on a UR5 industrial robotic platform, built upon the $\pi_{0.5}$ control framework. "PM" denotes **PredMind**.

Experimental Setup. We evaluate our method on a real-world industrial assembly platform. A diverse set of manipulation tasks is constructed to cover representative production scenarios. For each task, 100 demonstration trajectories are collected. Data from all tasks are jointly used for multi-task fine-tuning. During evaluation, each task is executed for 100 rollouts to obtain stable success rate.

Results. We further evaluate our method on a real-world industrial assembly platform with contact-rich and long-horizon manipulation tasks. As shown in Figure 5, our approach consistently outper-

forms the strong baseline $\pi_{0.5}$ across both tasks. For *move the steel sleeve to the tray*, our method achieves a success rate of 58%, compared to 42% for $\pi_{0.5}$. On *fix the metal body on the fixture table*, we obtain 48% versus 38%. The consistent improvements across transport and precise placement tasks indicate enhanced robustness and more stable long-horizon execution in real-world settings.

6 Conclusion and Limitations

This paper introduces **PredMind**, a future-aware training framework that injects predictive supervision into the semantic hierarchy of the VLA backbone without increasing inference-time computation. Experiments across simulation and real-robot tasks show that **PredMind** achieves state-of-the-art performance with substantial efficiency gains, suggesting a practical path toward more dynamic, robust, and data-efficient VLA policies.

Limitations. PredMind primarily studies short-horizon future alignment during VLA fine-tuning. Our results therefore support native predictive alignment, but do not constitute a complete recipe for general VLA world modeling or large-scale pretraining. Future work could explore how to incorporate such objectives more deeply into large-scale pretraining, how to leverage richer temporal contexts beyond short-horizon prediction, and whether predictive supervision should be explicitly conditioned on actions to better capture controllable dynamics.

Acknowledgments and Disclosure of Funding

This work was supported by the Brain Science and Brain-like Intelligence Technology—National Science and Technology Major Project (Grant No. 2022ZD0208800) and the National Natural Science Foundation of China (Grant Nos. 62495092 and 62495090).

The authors declare no competing interests.

References

[1] 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.

[2] 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.

[3] Rui Cai, Jun Guo, Xinze He, Piaopiao Jin, Jie Li, Bingxuan Lin, Futeng Liu, Wei Liu, Fei Ma, Kun Ma, Feng Qiu, Heng Qu, Yifei Su, Qiao Sun, Dong Wang, Donghao Wang, Yunhong Wang, Rujie Wu, Diyun Xiang, Yu Yang, Hangjun Ye, Yuan Zhang, and Quanyun Zhou. Xiaomi-robotics-0: An open-sourced vision-language-action model with real-time execution. *arXiv preprint arXiv:2602.12684*, 2026.

[4] 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.

[5] Shaoqi Dong, Chaoyou Fu, Haihan Gao, Yi-Fan Zhang, Chi Yan, Chu Wu, Xiaoyu Liu, Yunhang Shen, Jing Huo, Deqiang Jiang, Haoyu Cao, Yang Gao, Xing Sun, Ran He, and Caifeng Shan. Vita-vla: Efficiently teaching vision-language models to act via action expert distillation, 2025. URL `https://arxiv.org/abs/2510.09607`.

[6] Senyu Fei, Siyin Wang, Junhao Shi, Zihao Dai, Jikun Cai, Pengfang Qian, Li Ji, Xinzhe He, Shiduo Zhang, Zhaoye Fei, Jingjing Gong, Jinlan Fu, and Xipeng Qiu. Libero-plus: In-depth robustness analysis of vision-language-action models. *arXiv preprint arXiv:2510.13626*, 2025.

[7] Chongkai Gao, Zixuan Liu, Zhenghao Chi, Junshan Huang, Xin Fei, Yiwen Hou, Yuxuan Zhang, Yudi Lin, Zhirui Fang, and Lin Shao. VLA-OS: Structuring and dissecting planning representations and paradigms in vision-language-action models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL `https://openreview.net/forum?id=PQYazNKEYo`.

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

[9] Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

[10] 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.

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

[12] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4015–4026, 2023.

[13] Fuhao Li, Wenxuan Song, Han Zhao, Jingbo Wang, Pengxiang Ding, Donglin Wang, Long Zeng, and Haoang Li. Spatial forcing: Implicit spatial representation alignment for vision-language-action model. *arXiv preprint arXiv:2510.12276*, 2025.

[14] Minghui Lin, Pengxiang Ding, Shu Wang, Zifeng Zhuang, Yang Liu, Xinyang Tong, Wenxuan Song, Shangke Lyu, Siteng Huang, and Donglin Wang. Hif-vla: Hindsight, insight and foresight through motion representation for vision-language-action models. *arXiv preprint arXiv:2512.09928*, 2025.

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

[16] Qi Lv, Weijie Kong, Hao Li, Jia Zeng, Zherui Qiu, Delin Qu, Haoming Song, Qizhi Chen, Xiang Deng, and Jiangmiao Pang. F1: A vision-language-action model bridging understanding and generation to actions. *arXiv preprint arXiv:2509.06951*, 2025.

[17] Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. Robocasa: Large-scale simulation of everyday tasks for generalist robots. *arXiv preprint arXiv:2406.02523*, 2024.

[18] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[19] 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.

[20] 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.

[21] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[22] Jinghuan Shang, Karl Schmeckpeper, Brandon B May, Maria Vittoria Minniti, Tarik Kelestemur, David Watkins, and Laura Herlant. Theia: Distilling diverse vision foundation models for robot learning. *arXiv preprint arXiv:2407.20179*, 2024.

[23] Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andres Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Remi Cadene. Smolvla: A vision-language-action model for affordable and efficient robotics, 2025. URL `https://arxiv.org/abs/2506.01844`.

[24] StarVLA Community and Von Neumann Institute, HKUST. Starvla: A lego-like codebase for vision-language-action model developing. *arXiv preprint arXiv:2604.05014*, 2026.

[25] Lin Sun, Bin Xie, Yingfei Liu, Hao Shi, Tiancai Wang, and Jiale Cao. Geovla: Empowering 3d representations in vision-language-action models. *arXiv preprint arXiv:2508.09071*, 2025.

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

[27] 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.

[28] Yang Tian, Sizhe Yang, Jia Zeng, Ping Wang, Dahua Lin, Hao Dong, and Jiangmiao Pang. Predictive inverse dynamics models are scalable learners for robotic manipulation. *arXiv preprint arXiv:2412.15109*, 2024.

[29] Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. *arXiv preprint arXiv:2509.09372*, 2025.

[30] Yuqi Wang, Xinghang Li, Wenxuan Wang, Junbo Zhang, Yingyan Li, Yuntao Chen, Xinlong Wang, and Zhaoxiang Zhang. Unified vision-language-action model. *arXiv preprint arXiv:2506.19850*, 2025.

[31] John Won, Kyungmin Lee, Huiwon Jang, Dongyoung Kim, and Jinwoo Shin. Dual-stream diffusion for world-model augmented vision-language-action model. *arXiv preprint arXiv:2510.27607*, 2025.

[32] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10371–10381, 2024.

[33] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *Advances in Neural Information Processing Systems*, 37:21875–21911, 2024.

[34] Jianke Zhang, Yanjiang Guo, Yucheng Hu, Xiaoyu Chen, Xiang Zhu, and Jianyu Chen. Up-vla: A unified understanding and prediction model for embodied agent. *arXiv preprint arXiv:2501.18867*, 2025.

[35] Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, Fan Lu, He Wang, et al. Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge. *arXiv preprint arXiv:2507.04447*, 2025.

[36] Han Zhao, Jingbo Wang, Wenxuan Song, Shuai Chen, Yang Liu, Yan Wang, Haoang Li, and Donglin Wang. Frappe: Infusing world modeling into generalist policies via multiple future representation alignment. *arXiv preprint arXiv:2602.17259*, 2026.

[37] Qingqing Zhao, Yao Lu, Moo Jin Kim, Zipeng Fu, Zhuoyang Zhang, Yecheng Wu, Zhaoshuo Li, Qianli Ma, Song Han, Chelsea Finn, et al. Cot-vla: Visual chain-of-thought reasoning for vision-language-action models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 1702–1713, 2025.

[38] Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. TraceVLA: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=b1CVu9l5GO`.

[39] Ruijie Zheng, Jing Wang, Scott Reed, Johan Bjorck, Yu Fang, Fengyuan Hu, Joel Jang, Kaushil Kundalia, Zongyu Lin, Loic Magne, et al. Flare: Robot learning with implicit world modeling. *arXiv preprint arXiv:2505.15659*, 2025.

A More Implementation Details

We use task success rate as the primary evaluation metric, measured by the percentage of episodes in which the agent completes the task within the allowed horizon. We use OpenVLA-OFT [11] as the base model on LIBERO [15] and GR00T N1.6 [1] on RoboCasa [17]. For LIBERO, we initialize from the OpenVLA-7B checkpoint; all newly introduced modules are randomly initialized. Training uses a single NVIDIA H800 for 2 epochs, and evaluation is run on an NVIDIA A100, following the official benchmark protocols. Inputs include two images per timestep and proprioception. We fine-tune with batch size 8 and learning rate 5×10^{-4}. The learning rate follows linear warmup (5% of total steps) and cosine decay. We enable standard image augmentation and use LoRA with rank 32. For RoboCasa, We follow the official GR00T N1.6 fine-tuning recipe: initialize from the GR00T-N1.6-3B checkpoint, train on 24 RoboCasa GR1 tabletop tasks from the PhysicalAI-Robotics-GR00T-X-Embodiment-Sim dataset using 8×H100 with global batch size 1024, learning rate 1×10^{-4}, AdamW, linear warmup (5%). For the future alignment objective, we align with a one-step future offset. The auxiliary loss weight is $\lambda = 0.3$. The alignment loss is the sum of MSE and cosine similarity, applied throughout training. For real-robot experiments, we adopt $\pi_{0.5}$ [9] and use the same fine-tuning protocol and hyperparameters as described above. We adopt XR0 for LIBERO-Plus because it exhibits stronger zero-shot transfer performance than OpenVLA-OFT under this benchmark. For LIBERO-Plus, we report only zero-shot transfer results, as this setting more directly evaluates robustness and generalization under visual and environmental distribution shifts. For training XR-0, we follow the official reproduction protocol provided by the official repos. Appendix C.1 examines the effect of instruction visibility through a token-ordering ablation in XR-0.

Hardware-normalized training budget. We provide an additional clarification for the training-budget comparison used in the main text. The comparison is made in the OpenVLA-OFT/LIBERO-Long setting. The fully trained OpenVLA-OFT baseline follows the original 8-GPU, 150K-update recipe, while **PredMind** reaches comparable or higher LIBERO-Long success rate with 20K updates on a single GPU. We therefore measure the training budget in GPU-steps, defined as the number of optimizer updates multiplied by the number of GPUs used for training. Under this accounting, the baseline uses $150\text{K} \times 8 = 1.2\text{M}$ GPU-steps, whereas **PredMind** uses $20\text{K} \times 1 = 20\text{K}$ GPU-steps. This corresponds to a $60\times$ reduction, or equivalently 1/60 of the hardware-normalized GPU-step budget. This comparison is not intended as a wall-clock speed-up claim, since wall-clock time also depends on implementation, parallel efficiency, hardware type, and system-level overhead.

B Full RoboCasa-GR1 Results

In the main paper, we report a compact RoboCasa-GR1 table that includes the articulated-container closing group and the overall average. Here, we provide the full task-level results over all 24 RoboCasa-GR1 tabletop tasks. All models are jointly trained on the same 24-task suite. Baseline results are evaluated over 50 rollouts per task, while **PredMind** is evaluated over 100 rollouts per task for more stable estimation.

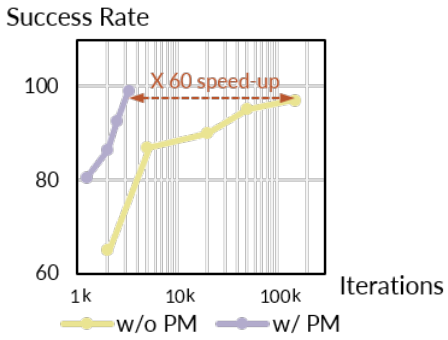

Figure 6: **Hardware-normalized training budget on OpenVLA-OFT/LIBERO-Long. PredMind** reaches comparable or higher success rate with 20K single-GPU updates, while the fully trained OpenVLA-OFT baseline uses 150K updates on 8 GPUs. This corresponds to a $60\times$ reduction in GPU-step budget.

Table 5: **Evaluation Results on RoboCasa GR1 Tabletop Tasks.** All models are jointly trained on all 24 tasks. Baselines report mean success rates over 50 rollouts per task; our method reports results over 100 rollouts per task for more reliable estimation.

Task	GR00T-N1.6	Qwen3-GR00T	Qwen3-PI	Qwen3-OFT	Qwen3-FAST	**PredMind (Ours)**
PnPBottleToCabinetClose	51.5	46.0	26.0	30.0	38.0	74.0
PnPCanToDrawerClose	13.0	80.0	62.0	76.0	44.0	80.0
PnPCupToDrawerClose	8.5	54.0	42.0	44.0	56.0	70.0
PnPMilkToMicrowaveClose	14.0	48.0	50.0	44.0	44.0	50.0
PnPPotatoToMicrowaveClose	41.5	28.0	42.0	32.0	14.0	40.0
PnPWineToCabinetClose	16.5	46.0	32.0	36.0	14.0	46.0
PnPNovelFromCuttingboardToBasket	58.0	48.0	40.0	50.0	54.0	50.0
PnPNovelFromCuttingboardToCardboardbox	46.5	40.0	46.0	40.0	42.0	40.0
PnPNovelFromCuttingboardToPan	68.5	68.0	60.0	70.0	58.0	64.0
PnPNovelFromCuttingboardToPot	65.0	52.0	40.0	54.0	58.0	58.0
PnPNovelFromCuttingboardToTieredbasket	46.5	56.0	44.0	38.0	40.0	58.0
PnPNovelFromPlacematToBasket	58.5	42.0	44.0	32.0	36.0	38.0
PnPNovelFromPlacematToBowl	57.5	44.0	52.0	58.0	38.0	50.0
PnPNovelFromPlacematToPlate	63.0	48.0	50.0	52.0	42.0	56.0
PnPNovelFromPlacematToTieredshelf	28.5	18.0	28.0	24.0	18.0	42.0
PnPNovelFromPlateToBowl	57.0	60.0	52.0	60.0	52.0	42.0
PnPNovelFromPlateToCardboardbox	43.5	50.0	40.0	50.0	30.0	52.0
PnPNovelFromPlateToPan	51.0	54.0	36.0	66.0	48.0	48.0
PnPNovelFromPlateToPlate	78.7	70.0	48.0	68.0	50.0	68.0
PnPNovelFromTrayToCardboardbox	51.5	38.0	34.0	44.0	28.0	60.0
PnPNovelFromTrayToPlate	71.0	56.0	64.0	56.0	34.0	64.0
PnPNovelFromTrayToPot	64.5	50.0	44.0	62.0	46.0	52.0
PnPNovelFromTrayToTieredbasket	57.0	36.0	50.0	54.0	36.0	38.0
PnPNovelFromTrayToTieredshelf	31.5	16.0	28.0	30.0	16.0	20.0
Average	47.6	47.8	43.9	48.8	39.0	**52.5**

C Additional Analysis

C.1 Effect of Language Conditioning

To examine the role of language-goal information in predictive representation learning, we compare two token orderings in XR-0 under a causal attention mask on LIBERO-Plus. When language tokens precede visual tokens, the visual representations can attend to the instruction. Reversing this order prevents visual tokens from accessing the instruction. Here, "with" and "without" language conditioning refer specifically to this visibility of instruction tokens to visual tokens.

Table 6: Success rates (%) on LIBERO-Plus under different language-conditioning configurations in XR-0. With language conditioning, language tokens precede visual tokens under a causal attention mask; without language conditioning, the token order is reversed. The XR-0 baseline is included for reference.

Method	Camera	Robot	Language	Light	Background	Noise	Layout
XR-0	40.0	52.3	84.9	98.0	92.6	82.6	78.5
+ PredMind (without language)	43.4	54.7	85.3	96.3	94.3	86.9	80.2
+ PredMind (with language)	69.4	60.5	88.1	94.9	96.8	91.8	82.7

As shown in Table 6, enabling access to language instructions improves six of the seven perturbation settings relative to the PredMind variant without language conditioning, although performance under lighting perturbations decreases. These results support the benefit of language-goal information for predictive representation learning.

C.2 Representation-Level Evidence

To examine how well the predictive objective is fitted during training, we record the predictive alignment loss $\mathcal{L}_{\mathrm{pm}}$ at several training stages. As shown in Table 7, the loss decreases from 0.996 at initialization to 0.011 at convergence, indicating improved fitting of the future-representation prediction objective.

Table 7: Predictive alignment loss at selected training stages.

Training stage	Initialization	5K steps	20K steps	Convergence
$\mathcal{L}_{\mathrm{pm}}$	0.996	0.031	0.018	0.011

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state the paper's main contribution as predictive alignment in the native VLM latent space for VLA policy learning. The claimed improvements are scoped to the evaluated simulation benchmarks and are supported by the experimental results.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of our approach.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper does not present formal theoretical results, theorems, or proofs. The method is an empirical algorithmic contribution evaluated through simulation experiments.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: We provide the training recipes, datasets, model backbones, optimization settings, evaluation protocols, and rollout counts needed to reproduce the main empirical findings.

Guidelines:

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

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

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

   Answer: [Yes]

   We provide the codebase in the supplementary material, including the core implementation, configuration files, and reproduction instructions for the main simulation experiments.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper specifies the training and evaluation datasets, model backbones, input modalities, optimization hyperparameters, LoRA configuration, rollout protocols, and baseline comparison settings.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: We do not report error bars for all experiments because full multi-seed training is computationally expensive for the evaluated VLA backbones. We instead report the number of evaluation rollouts per task and use the same evaluation protocol across methods.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We report the GPU types, number of devices, batch sizes, training duration, evaluation hardware.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and believe the research conforms to it. The experiments use existing simulation benchmarks and pretrained models under their respective terms, and do not involve human subjects, private data, or deceptive data collection.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

Justification: The work is evaluated through simulation benchmarks and real-world robotic manipulation experiments and does not release a deployable robot system.

Guidelines:

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

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper does not release high-risk pretrained generative models, scraped datasets, or real-world robot control systems. The evaluated policies are studied in simulation benchmarks, and no broadly deployable physical robot system is released.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We cite the original papers and repositories for all existing datasets, simulators, pretrained models, and baseline implementations used in this work.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.

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

13. **New assets**

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

    Answer: [Yes]

    Justification: We provide code as a new research asset in the supplementary material, together with documentation including installation instructions, training and evaluation commands, configuration files, and license information.

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing, user studies, or research with human subjects. Experiments use pretrained models and include both simulation benchmarks and real-world robotic manipulation tasks.

    Guidelines:

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

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

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

Answer: [N/A]

Justification: The paper does not involve crowdsourcing, user studies, or human-subject research, so IRB or equivalent approval is not applicable.

Guidelines:

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

16. **Declaration of LLM usage**

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

Justification: LLMs were not used as an important, original, or non-standard component of the core research methodology.

Guidelines:

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