CONTROLLING ACTION-HEAD CO-ADAPTATION TO VLM REPRESENTATION ANISOTROPY

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern learning systems increasingly combine pretrained models with downstream heads. Despite the growing importance of such modular designs, how the downstream model co-adapts to the geometry of pretrained representations remains relatively underexplored. We study this problem through vision-language-action models (VLAs), where a pretrained vision-language model (VLM) conditions an action model for control. We find that the action model progressively concentrates its sensitivity on high-variance dimensions of the VLM outputs. Through controlled experiments across different backbone-head models, we further show that counteracting this sensitivity concentration can be beneficial under representation–task mismatch settings. Motivated by these observations, we introduce *Adaptive Interface Equalizer* (AdaIE), a parameter-free mechanism that reweights pretrained representations according to the evolving downstream sensitivity during training. AdaIE improves VLA success rates on simulation benchmarks, with gains of up to 5.28% on RoboCasa-GR1 and 1.68% on RoboTwin 2.0, and achieves an average gain of 19% across five real-world manipulation tasks. It further generalizes to diffusion policies and world models, improving downstream performance in both settings. Our results highlight interface co-adaptation as an important optimization problem when coupling pretrained representations with downstream models.

1 INTRODUCTION

VLAs transfer pretrained VLM knowledge to robotic control (Ma et al., 2026; Kim et al., 2024). Recent approaches use diffusion- or flow-matching-based action heads to map VLM representations of observations and instructions to continuous actions (Black et al., 2024; Li et al., 2024; Chi et al., 2025; Bjorck et al., 2025). Despite substantial progress in data pipelines, architectures, and training (Brohan et al., 2023; Intelligence et al., 2025; Hou et al., 2025; Li et al., 2026), the representation interface connecting VLMs and action heads remains comparatively underexplored.

The VLM and action model are typically connected through a lightweight conditioning interface, such as cross-attention and concatenation. Through this interface, representations produced by the VLM are directly consumed by the action model for action learning. Despite its simplicity, this interface determines how the representation geometry learned from vision-language pretraining is exposed to the action head.

In this work, begin by studying this interface in VLAs, we find that VLM outputs exhibit strong channel-wise anisotropy, and that the action model progressively becomes more sensitive to these dimensions during training (See Figure 1). However, the two components are optimized for different objectives: the VLM is pretrained to organize information for vision-language understanding, whereas the action head is trained for control. We therefore ask:

Why should an action head trained for control increasingly rely on the same feature dimensions shaped by a VLM pretrained for a different objective?

To answer this question, we study the interface by tracking VLM channel variance and action-head sensitivity throughout training, and further compare sensitivity reweighting across interfaces with different degrees of representation–task alignment. We find that counteracting the evolving sensitivity imbalance consistently improves performance under representation–task mismatch.

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

Motivated by these observations, we introduce *Adaptive Interface Equalizer* (AdaIE), a parameter-free mechanism that reweights representations using evolving downstream sensitivity. We summarize our contributions as follows:

- We identify backbone–head co-adaptation: the head progressively concentrates its sensitivity on high-variance dimensions of pretrained backbone representations. We show that reducing sensitivity concentration can improve learning under representation–task mismatch.
- We propose *Adaptive Interface Equalizer* (AdaIE), a parameter-free mechanism that adaptively reweights backbone representations according to the evolving head sensitivity to regulate interface co-adaptation.
- We demonstrate AdaIE's effectiveness in VLAs, with success-rate gains of up to 5.28% on RoboCasa-GR1, 1.68% on RoboTwin 2.0, and an average of 19% across real-world manipulation tasks. We further show that its benefits extend to diffusion policies and world models.

2 Action-Head Co-Adaptation to VLM Anisotropy

We begin by examining the VLM–DiT interface, characterizing VLM channel-wise anisotropy and the evolution of action-head sensitivity throughout training in Section 2.1. We then investigate whether this emerging co-adaptation is beneficial for downstreaming learning in Section 2.2.

2.1 VLM Representation Anisotropy and Action-Head Co-Adaptation

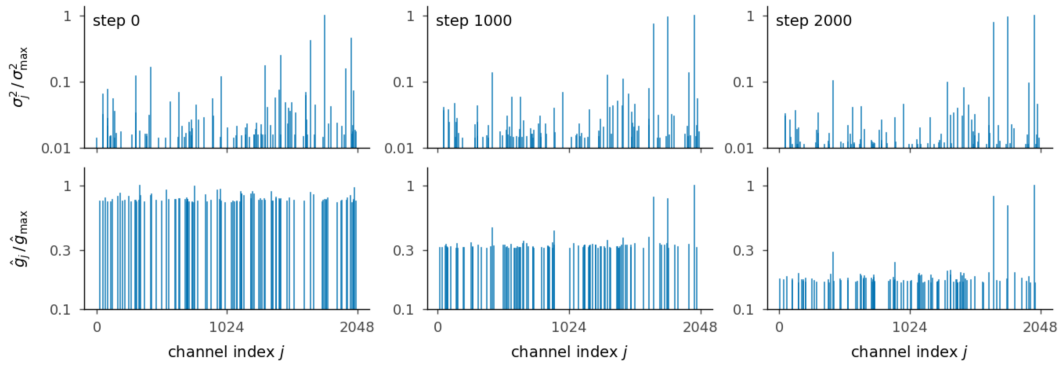


Figure 1: Action-head sensitivity progressively aligns with the anisotropic VLM representation during training. **Top**: per-channel variance of the VLM output dimensions at different training steps. **Bottom**: corresponding per-channel action-head sensitivity. While sensitivity is relatively uniform early in training, it progressively concentrates on high-variance VLM dimensions as training proceeds, revealing increasing alignment between VLM anisotropy and action-head sensitivity.

We first introduce the notation used throughout this section. Let $\mathbf{C} \in \mathbb{R}^{N \times D}$ denote the VLM conditioning tokens, where N is the number of tokens and D is the hidden dimension. We use $i \in \{1, \ldots, N\}$ to index conditioning tokens and $j \in \{1, \ldots, D\}$ to index feature channels, such that $\mathbf{C}_{i,j}$ denotes the activation of channel j at token i. Let $\mathbf{X} \in \mathbb{R}^{N \times D}$ denote input tokens of the action DiT. In the standard VLAs, $\mathbf{X} = \mathbf{C}$.

With this notation, we track two per-channel statistics throughout training: the variance of the VLM representation and the sensitivity of the action head to each conditioning channel.

VLM anisotropy. We characterize the anisotropy of VLM representations via per-channel variance,

$$\sigma_j^2 = \mathrm{Var}[\mathbf{C}_{:,j}].$$

The variance is estimated over conditioning tokens. A highly uneven variance across channels indicates an anisotropic representation, where a small subset of dimensions varies more than the rest.

Head sensitivity. We characterize the sensitivity of the head model (e.g. DiT action head) to feature channel j of its input $\mathbf{X}$ by averaging its sensitivity across conditioning tokens. For token i, let

$$\mathbf{J}_i = \frac{\partial \hat{\mathbf{a}}}{\partial \mathbf{X}_{i,:}}$$

denote the Jacobian of the action prediction with respect to its D feature channels. We define the sensitivity of channel j as

$$\mathbf{g}_j = \mathbb{E}_i \Big[\mathrm{diag}\big(\mathbf{J}_i^\top \mathbf{J}_i\big)_j \Big],$$

where the expectation averages over conditioning tokens i. To estimate $\mathbf{g_j}$ efficiently, we use Hutchinson-style stochastic probing (Hutchinson, 1989)(Algorithm 1).

To study those two metrics, we adopt a VLM–DiT architecture following GR00T-N1 (Bjorck et al., 2025), where vision-language representations condition a flow-matching DiT action head through cross-attention. Following the RoboCasa-GR1 training setup in StarVLA (Community, 2026), we use a pretrained Qwen3-VL (Bai et al., 2025) as the VLM backbone and train the action DiT from scratch. For clarity, Figure 1 visualizes the top-100 VLM channels with the highest variance.

As shown in Figure 1, the VLM exhibits a strongly anisotropic channel-wise variance, while the action-head sensitivity is relatively uniform early in training. As the training goes, the action head becomes increasingly sensitive to the same high-variance VLM dimensions. This growing alignment suggests that the action head progressively co-adapts to the anisotropic structure of the pretrained VLM representation. A similar phenomenon is also observed in other models (see Appendix A.6).

2.2 Is Co-Adaptation Beneficial for Action Learning?

The emerging co-adaptation has two opposing interpretations. One is that the action head becomes more sensitive to high-variance VLM dimensions because these dimensions are more useful for downstream control. An alternative view is that the action head is increasingly shaped by the anisotropic geometry inherited from vision-language pretraining, which may not be optimal for action learning because the VLM and action model are optimized for different domains and objectives. Under this view, reducing the emerging sensitivity imbalance may instead be beneficial.

To distinguish between these two interpretations, we construct a controlled experiment on the sensitivity. We compare *Exploit*, which reinforces the emerging sensitivity pattern, with *Equalize*, which reduces it, using the original training as the *Baseline*.

Specifically, given the estimated sensitivity $\hat{\mathbf{g}}_\mathbf{j}$, we assign each channel a sensitivity-dependent weight

$$\mathbf{s}_j = (\hat{\mathbf{g}}_j)^{-\alpha}, \tag{1}$$

where $\alpha > 0$ gives *Equalize*, assigning relatively smaller weights to more-sensitive channels, while $\alpha < 0$ gives *Exploit*, assigning relatively larger weights to them.

To examine when such co-adaptation is beneficial or harmful, we consider two types of backbone–head interfaces with different degrees of representation–task alignment:

- **Aligned.** The pretrained representation is expected to be well suited to the downstream task. We consider (1) DINOv2-S with a DPT head (Yang et al., 2024) on NYUv2 (Silberman et al., 2012) for depth estimation, and (2) DeiT (Touvron et al., 2021) on CIFAR-100 (Krizhevsky, 2009) for image classification.
- **Mismatched.** The pretrained representation is learned for an objective substantially different from downstream action generation. We consider (1) Diffusion Policy (Chi et al., 2025) on Push-T for visuomotor action learning, and (2) GR00T (Bjorck et al., 2025) on RoboCasa-GR1 (Nasiriany et al., 2024) for vision-language-action learning.

To study how downstream heads co-adapt to pretrained representations, we initialize the backbone in each of the four tasks with pretrained weights and otherwise follow the official training recipe. Within each task, all compared variants use the same configuration, differing only in the value of α. Detailed training setups are provided in Appendix A.4.

Table 1: Effect of sensitivity reweighting under aligned and mismatched interfaces.

Task	Dataset	Interface	Metric	Equalize	Baseline	Exploit
Aligned						
Depth	NYUv2	DINOv2-S → DPT	δ_1 (%) ↑	93.4± 0.3	93.5± 0.1	93.2± 0.3
Classification	CIFAR-100	DeiT-S → Cls.	Acc. (%)↑	90.0 ± 0.2	89.6 ± 0.4	89.8 ± 0.4
Mismatched						
Diffusion Policy	Push-T	ResNet → DP	Score↑	0.936 ± 0.003	0.922 ± 0.003	0.909 ± 0.009
VLA	RoboCasa	VLM → DiT	Succ. (%)↑	22.9 ± 0.5	19.4 ± 0.6	16.2 ± 1.1

The results are shown in Table 1. For the aligned tasks, sensitivity reweighting has only a marginal effect on performance, with all three variants performing similarly. While in the mismatched tasks exhibit a clear and consistent trend: *Equalize* outperforms the original *Baseline*, while *Exploit* degrades performance. This contrast indicates that co-adaptation to the pretrained representation may be more problematic when the representation is less aligned with the downstream objective.

Motivated by the observations above, we propose *Adaptive Interface Equalizer* (AdaIE) to improve backbone–head training under representation–task mismatch.

3 Adaptive Interface Equalizer

We now introduce *Adaptive Interface Equalizer* (AdaIE), a parameter-free mechanism that continuously compensates for the evolving sensitivity imbalance during training. AdaIE consists of two steps:sensitivity estimation (Section 3.1) and sensitivity equalization (Section 3.2). We adopt VLA as the representive model to introduce AdaIE.

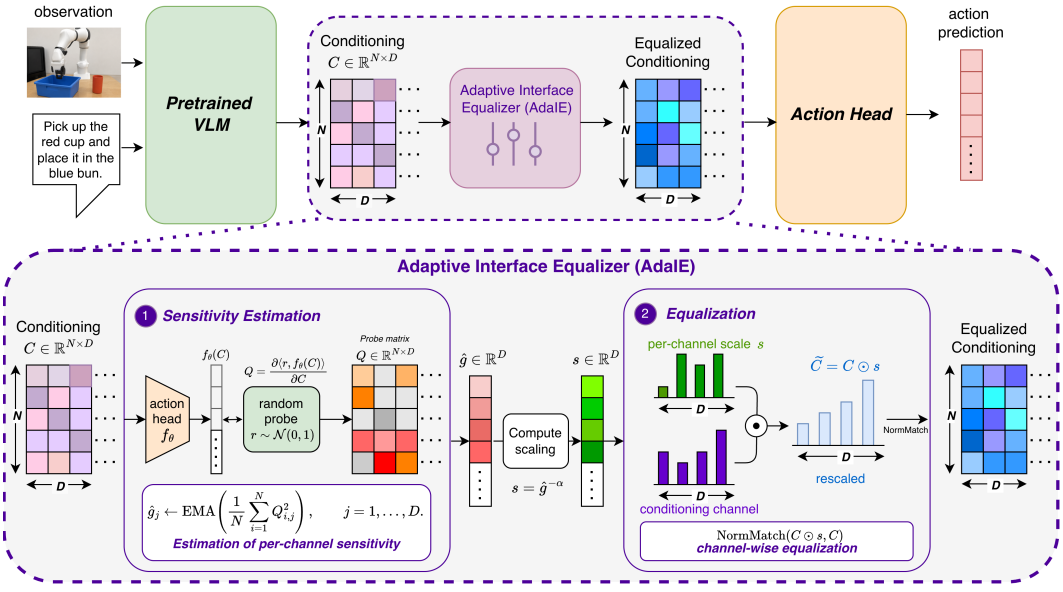


Figure 2: Overview of Adaptive Interface Equalizer (AdaIE) on VLAs. AdaIE operates at the interface between the VLM and the action DiT. It estimates the action head's per-channel sensitivity online using stochastic probes, maintains a running sensitivity estimate, and inversely rescales the VLM conditioning channels before they are consumed by the DiT.

3.1 Sensitivity Estimation

Let $\mathbf{C} \in \mathbb{R}^{N \times D}$ denote the VLM conditioning at the VLM–DiT interface. As defined in Section 2.1, we measure the DiT sensitivity of channel j by the squared Jacobian norm of the action prediction

with respect to that channel. Computing the full Jacobian is prohibitively expensive, so we estimate its diagonal online using Hutchinson's random probe (Hutchinson, 1989). Specifically, for a random output-space direction $\mathbf{r} \sim \mathcal{N}(0, I)$ and a action head f_θ, we compute the *sensitivity probe matrix*

$$\mathbf{Q} = \frac{\partial \langle \mathbf{r}, f_\theta(\mathbf{C}) \rangle}{\partial \mathbf{C}} \in \mathbb{R}^{N \times D}. \tag{2}$$

The per-channel sensitivity estimate is

$$\mathbf{g}_j = \frac{1}{N} \sum_{i=1}^{N} \mathbf{Q}_{i,j}^2, \tag{3}$$

which is then exponentially averaged during training to obtain $\hat{g}_j$. Algorithm 1 summarizes the full process. In practice, the probe is performed every K optimization steps rather than at every step, introducing little additional training overhead.

Algorithm 1 Sensitivity Estimation

Require: VLM conditioning $\mathbf{C} \in \mathbb{R}^{N \times D}$; action head f_θ; EMA sensitivity $\hat{\mathbf{g}} \in \mathbb{R}^D$
Ensure: Updated sensitivity $\hat{\mathbf{g}}$

$r \sim \mathcal{N}(0, I)$ ▷ random output-space probe
$\mathbf{Q} \leftarrow \frac{\partial \langle r, f_\theta(\mathbf{C}) \rangle}{\partial \mathbf{C}}$ ▷ sensitivity probe matrix
$\mathbf{g} \leftarrow \mathrm{mean}_{i=1}^{N}(\mathbf{Q}_i^2)$ ▷ per-channel sensitivity
$\hat{\mathbf{g}} \leftarrow \rho\, \hat{\mathbf{g}} + (1 - \rho)\, \mathbf{g}$ ▷ EMA over training
return $\hat{\mathbf{g}}$

3.2 Sensitivity Equalization

Given the estimated sensitivity $\hat{\mathbf{g}}_j$, AdaIE assigns each conditioning channel a sensitivity-dependent scaling factor:

$$\mathbf{s}_j = (\hat{\mathbf{g}}_j)^{-\alpha}, \qquad \widetilde{\mathbf{C}}_{i,j} = \mathbf{C}_{i,j} \mathbf{s}_j, \tag{4}$$

where $\alpha \geq 0$ controls the equalization strength. Larger α produces stronger equalization, while $\alpha = 0$ recovers the original conditioning.

Channel-wise rescaling can change the overall magnitude of the conditioning. We therefore match the magnitude of the transformed representation to that of the original VLM output:

$$\mathbf{Z} = \mathrm{NormMatch}(\widetilde{\mathbf{C}}, \mathbf{C}). \tag{5}$$

This removes the global scale change introduced by Eq. 4, leaving only the relative reweighting across channels. In AdaIE, we apply Algorithm 1 to the reweighted and norm-matched conditioning $\mathbf{Z}$, and the scaling factors for each forward pass are computed from the EMA of preceding sensitivity probes. We summarize AdaIE in Figure 2 and Algorithm 2. The transformation introduces no learnable parameters, and its sensitivity statistics are updated online jointly with VLA training, and frozen during inference.

Algorithm 2 Adaptive Interface Equalizer (AdaIE)

Require: VLM output $\mathbf{C} \in \mathbb{R}^{N \times D}$; sensitivity $\hat{\mathbf{g}} \in \mathbb{R}^D$; equalization strength α; stabilization constant ϵ
Ensure: DiT conditioning $\mathbf{Z} \in \mathbb{R}^{N \times D}$

$s \leftarrow (\hat{g} + \epsilon)^{-\alpha}$ ▷ sensitivity-based scaling
$\widetilde{\mathbf{C}} \leftarrow \mathbf{C} \odot s$ ▷ channel-wise reweighting
$\mathbf{Z} \leftarrow \mathrm{NormMatch}(\widetilde{\mathbf{C}}, \mathbf{C})$ ▷ preserve conditioning magnitude
return $\mathbf{Z}$

4 Experiments

We first evaluate AdaIE on VLA models in both simulation and real-world manipulation settings. In simulation, we adopt RoboCasa-GR1 (Nasiriany et al., 2024) and RoboTwin 2.0 (Chen et al., 2025),

Table 2: Average success rate on RoboCasa-GR1 (24 tasks). Results above the separator are reported by StarVLA (Community, 2026), while results below are our runs, evaluated with 60 rollouts per task. Detailed per-task results of our runs are provided in Appendix A.5.

Method	Backbone	Steps	Avg. (%)
$\pi_{0.5}$	–	–	37.00
GR00T-N1.6	–	–	47.60
StarVLA-GR00T	Qwen3-VL-4B	100K	47.80
StarVLA-π	Qwen3-VL-4B	100K	43.90
StarVLA-OFT	Qwen3-VL-4B	100K	48.80
GR00T	Qwen3-VL-2B	100K	42.36
GR00T + AdaIE	Qwen3-VL-2B	100K	**47.64** (+5.28)
GR00T	Qwen3-VL-4B	100K	47.99
GR00T + AdaIE	Qwen3-VL-4B	100K	**50.07** (+2.08)

Table 3: Success rates on RoboTwin 2.0 (50 tasks). Both methods are trained for 150K optimization steps using identical training configurations. See per task comparison in the Appendix A.5.

Method	Backbone	Steps	Success Rate (%)
GR00T	Cosmos-Reason2-2B	150K	66.92
GR00T + AdaIE	Cosmos-Reason2-2B	150K	**68.60**(+1.68)

following the settings of StarVLA (Community, 2026). For real-world evaluation, we train and deploy the policy on IRON-R01-1.11 humanoid robots across diverse physical manipulation tasks. To examine whether AdaIE is limited to VLAs, we further extend it to world models, evaluating LeWM (Maes et al., 2026) and DINO-WM (Zhou et al., 2024) on multiple control tasks.

4.1 Robotic Manipulation Benchmarks

4.1.1 Simulation Benchmarks

Following StarVLA (Community, 2026), we use identical training settings across variants, varying only α. Full configurations are provided in Appendix A.1.

RoboCasa-GR1. RoboCasa-GR1 (Nasiriany et al., 2024) is a tabletop manipulation benchmark built on the RoboCasa simulation framework and is commonly used to evaluate humanoid manipulation policies. The benchmark contains 24 tasks, with approximately 1,000 demonstrations per task, resulting in about 24K trajectories in total. We adopt a VLM–DiT architecture following GR00T-N1 (Bjorck et al., 2025), with a pretrained Qwen3-VL backbone and a DiT action head from scratch. We train all policies with a batch size of 256 for 100K optimization steps. For evaluation, we run 60 rollouts per task.

As shown in Table 2. AdaIE improves the baseline at both model scales. With the 2B backbone, AdaIE improves the success rate from 42.36% to 47.64%, and achieves performance comparable to the results of 4B backbone. The improvement persists when scaling the backbone to 4B.

RoboTwin 2.0. RoboTwin 2.0 (Chen et al., 2025) is a large-scale benchmark for bimanual robotic manipulation, covering 50 tasks under both clean and randomized settings. Each task provides 50 clean and 500 randomized demonstrations, resulting in approximately 27.5K trajectories in total.

Following the setup in StarVLA, we train the policy for 150K optimization steps with a global batch size of 192. For evaluation, we follow StarVLA and run 100 episodes per task, resulting in 10,000 evaluation episodes in total. The results are shown in Table 3.

Training overhead. AdaIE introduces only modest additional training cost. On the 2B model using $4\times$H200 GPUs for 150K training steps, the baseline requires 65.45 hours, while AdaIE requires 67.88 hours, corresponding to a 3.7% increase in wall-clock training time.

4.1.2 Real-world Robots Manipulation

To evaluate AdaIE under real physical interaction, we deploy the policy on an IRON-R01-1.11 dual-arm humanoid robot. The policy receives a single head-mounted fisheye observation, a language instruction, and the current proprioceptive state, and predicts action chunks for manipulation.



Figure 3: Real-world manipulation evaluation. For each task, we conduct 10 trials with the left hand and 10 trials with the right hand. See Table 4 for the detailed results.

Data and Training. We train on a curated real-robot dataset collected from five humanoid robots, comprising 5,875 teleoperated episodes across 21 manipulation task categories. The original 50-Hz trajectories are sampled at 10 Hz. Each training sample consists of an RGB observation, a language instruction, the current robot state, and a 30-step action chunk.

We adopt the same GR00T architecture as in simulation, using a pretrained Qwen3-VL-2B-Instruct backbone and a DiT action head trained from scratch. The baseline and AdaIE variants use exactly the same training data, optimization schedule, and model architecture, differing only by the insertion of AdaIE. Full dataset composition and training details are provided in Appendix A.2.

Evaluation. We evaluate the baseline and AdaIE on 5 real-world manipulation tasks. Each method is evaluated for 20 trials (10 trials for left hand, and 10 for right hand) per task under identical initial conditions and task instructions. A trial is considered successful if the robot completes the specified task objective according to a predefined task-specific criterion, without human intervention. We additionally report Interaction Success, which measures whether the robot successfully reaches and interacts with the target object or mechanism, even if the full task objective is not completed. For example, in pick-and-place tasks, a trial is counted as an interaction success if the robot successfully grasps the target object but fails to place it in the designated tray. The results are shown in Table 4, where AdaIE improves the overall task success rate compares to the baseline model.

4.2 Extension to World Models

AdaIE is not limited to VLA architectures and can be applied to more general cross-model interfaces. We further apply AdaIE to a JEPA-based (LeCun et al., 2022) world-model framework, using LeWM (Maes et al., 2026) and DINO-WM(Zhou et al., 2024) as representative instantiations.

LeWorldModel. Following the experimental setup of LeWM (Maes et al., 2026), we train both the original LeWM baselines and their AdaIE variants on four tasks using identical configurations. LeWM is an end-to-end JEPA consisting of a ViT encoder that maps image observations into compact latent representations and a predictor that predicts future latent states. The encoder and predictor are jointly optimized with a latent prediction objective. We apply AdaIE to the encoder representation before it is consumed by the predictor.

Table 4: Real-robot evaluation results on in-distribution (ID) and out-of-distribution (OOD) tasks.

		Baseline		AdaIE	
Task	**Hand**	**Task Succ.**	**Interaction Succ.**	**Task Succ.**	**Interaction Succ.**
In-Distribution (ID)					
Banana	Left	5/10	10/10	**6/10**	9/10
	Right	4/10	10/10	**7/10**	10/10
Press Bell	Left	**6/10**	9/10	5/10	10/10
	Right	2/10	9/10	**4/10**	8/10
ID Overall	–	42.5%	**95.0%**	**55.0%** (+12.5%)	92.5% (-2.5%)
Out-of-Distribution (OOD)					
Toy Car	Left	4/10	10/10	**5/10**	10/10
	Right	2/10	9/10	**5/10**	10/10
Cheese Ball	Left	2/10	7/10	**3/10**	7/10
	Right	2/10	5/10	**5/10**	8/10
Milk Bottle	Left	2/10	7/10	**5/10**	8/10
	Right	1/10	8/10	**4/10**	10/10
OOD Overall	–	21.7%	76.7%	**45.0%** (+23.3%)	**88.3%** (+11.7%)
Overall	–	30.0%	84.0%	**49.0%** (+19.0%)	**90.0%** (+6.0%)

Task	Method	Success Rate (%)
Two Room	LeWM	91.6 ± 2.0
	LeWM + Var.	91.2 ± 1.6
	LeWM + AdaIE	$\mathbf{93.2 \pm 1.0}$
Push-T	LeWM	92.8 ± 3.7
	LeWM + Var.	93.6 ± 1.5
	LeWM + AdaIE	$\mathbf{95.6 \pm 1.5}$
Reacher	LeWM	76.4 ± 3.7
	LeWM + Var.	76.8 ± 4.1
	LeWM + AdaIE	$\mathbf{78.4 \pm 5.9}$
OGBench-Cube	LeWM	76.4 ± 2.3
	LeWM + Var.	$\mathbf{77.2 \pm 1.0}$
	LeWM + AdaIE	76.8 ± 3.2

Table 5: Planning performance of LeWM and LeWM + AdaIE. The final planning success rate is reported. For each model, we follow the eval policy of LeWM (Maes et al., 2026), with 50 episodes per task. We use $\alpha = 0.25$ for AdaIE.

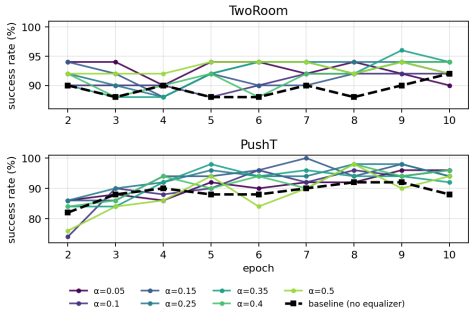


Figure 4: Ablation of $\alpha \in \{0.05, \cdots, 0.5\}$ on LeWM. We report evaluation success rates across training epochs. Full quantitative results are provided in Table 15.

The LeWM evaluation contains four tasks: Two Room, Push-T, Reacher, and OGBench-Cube, covering 2D navigation, 2D manipulation and control, and 3D robotic manipulation. We run 5 random seed training for all variants for LeWM, and report mean± standard deviation over five training seeds in Table 5. See Appendix A.3 for additional experimental details.

Ablation on α . We further study the effect of α. As shown in Figure 4, most α values improve planning performance over the baseline, although some settings are slightly worse than the baseline. The best α also differs across tasks, and how to derive the reweighting strength adaptively without task-specific tuning remains an open question. We provide the full per-epoch results in Table 15.

Sensitivity vs. variance. We further test whether downstream sensitivity can be replaced by the channel-wise variance of the pretrained representation. To test this, we replace the sensitivity estimate $\hat{g}_j$ with the channel variance $\hat{\sigma}_j^2$, while keeping the rest of the reweighting pipeline identical. As shown in Table 5, variance-based reweighting also yields gains on several tasks, while AdaIE performs the best in three of four over them.

Table 6: Performance of DINO-WM and DINO-WM + AdaIE.

Push-T			Reacher		
Method	**Success (%)**	**Std.**	**Method**	**Success (%)**	**Std.**
DINO-WM	71.4	7.0	DINO-WM	78.6	3.9
DINO-WM + AdaIE	**73.2** (+1.8)	6.5	DINO-WM + AdaIE	**82.8** (+4.2)	6.3

DINO-WM. We further evaluate AdaIE on DINO-WM (Zhou et al., 2024). Following the DINO-WM setting adopted by LeWM (Maes et al., 2026), we train all models for 10 epochs. For evaluation, we follow the same model-predictive planning protocol as LeWM and report mean success rate over 10 random seeds, with 50 episodes per seed. The results are reported in Table 6.

5 Related Work

5.1 Vision-Language-Action Models

Vision-language-action (VLA) models integrate visual observations, language instructions, and robot actions into a unified policy. Early approaches such as RT-1 and RT-2 (Brohan et al., 2022; 2023) formulate action generation autoregressively, while more recent systems increasingly adopt continuous-action diffusion or flow-matching policies, including π_0 (Black et al., 2024) and GR00T N1 (Bjorck et al., 2025). Open-source models such as Octo (Mees et al., 2024) and OpenVLA (Kim et al., 2024) further demonstrate the effectiveness of transferring pretrained vision-language representations to robotic control. Many follow-up works improve these frameworks through action-model design (Li et al., 2024), efficient fine-tuning and action generation (Kim et al., 2025; Pertsch et al., 2025), and large-scale cross-task and cross-embodiment training (Intelligence et al., 2025).

5.2 World Models

World models learn representations of environment dynamics to predict future states and support downstream decision-making and planning. Early approaches learn latent dynamics for model-based control (Ha & Schmidhuber, 2018; Hafner et al., 2019), recent methods explore predictive representation learning, such as JEPA (LeCun et al., 2022), and pretrained visual representations for dynamics prediction. DINO-WM (Zhou et al., 2024) and LeWM (Maes et al., 2026), for example, predict future states in representation space and use the learned dynamics for downstream planning. Many other following works explore the optimization of those frameworks, including training (Agarwal et al., 2026; Kang et al., 2026), planning (Psenka et al., 2026) and others (Zhang et al., 2026; Klindt et al., 2026; Luo et al., 2026; Rao et al., 2026).

6 Conclusion and Limitations

We uncover *interface co-adaptation*: downstream models progressively adapt their sensitivity to the anisotropic geometry of the backbone representations. Our observations suggest that counteracting sensitivity concentration can benefit downstream learning. We introduce AdaIE to regulate it through sensitivity-based reweighting. Across VLAs, diffusion policies, and world models, AdaIE consistently improves downstream performance. More broadly, our results establish the representation interface itself as a important optimization problem in modular learning systems.

Limitations. Our aligned/mismatched grouping in Section 2.2 reflects an a priori assumption that pretraining and downstream objectives are more closely aligned in depth estimation and classification than in our robot-control settings. Quantifying this alignment and establishing its causal role remain open questions. Due to computational resource constraints, we are only able to conduct multi-seed training for a subset of our experiments. AdaIE requires periodic sensitivity estimation, which introduces additional training cost. Our experiments cover VLAs, diffusion policies, and world models, but not broader multimodal architectures. The current equalizer also operates on single-layer conditioning, leaving multi-layer interfaces (e.g., Mixture of Transformers) for future work. Finally, the strength α may require tuning across tasks and architectures.

AI USE STATEMENT

In this work, we used generative AI tools to provide feedback on experiments, assist with dataset reformatting, summarize quantitative experimental results, and implement parts of the AdaIE integration into the evaluated model. Additionally, we use these tools to improve the clarity and readability of the manuscript. We have reviewed all AI-assisted work and checked the implementation correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our real-robot experiments use company-owned robots and proprietary demonstration data. Physical testing is confined to a restricted operating area to mitigate safety risks. Any release of proprietary data or platform-specific assets is subject to company authorization.

REPRODUCIBILITY STATEMENT

Our method is described in Section 3, Algorithms 1 and 2. Training configurations and evaluation protocols are provided in Appendix Sections A.1, A.2, A.3, and A.4, with per-task results in Appendix A.5. We plan to release our implementation. Full reproduction of the real-robot experiments requires access to proprietary data and company-owned hardware, subject to company authorization.

REFERENCES

Niket Agarwal, Arslan Ali, Jon Allen, Martin Antolini, Adeline Aubame, Alisson Azzolini, Junjie Bai, Maciej Bala, Yogesh Balaji, Josh Bapst, et al. Cosmos 3: Omnimodal world models for physical ai. *arXiv preprint arXiv:2606.02800*, 2026.

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.

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

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. $\pi 0$: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Tianxing Chen, Zanxin Chen, Baijun Chen, Zijian Cai, Yibin Liu, Zixuan Li, Qiwei Liang, Xianliang Lin, Yiheng Ge, Zhenyu Gu, et al. Robotwin 2.0: A scalable data generator and benchmark with strong domain randomization for robust bimanual robotic manipulation. *arXiv preprint arXiv:2506.18088*, 2025.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 44(10-11):1684–1704, 2025.

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

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2(3):440, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Zhi Hou, Tianyi Zhang, Yuwen Xiong, Haonan Duan, Hengjun Pu, Ronglei Tong, Chengyang Zhao, Xizhou Zhu, Yu Qiao, Jifeng Dai, et al. Dita: Scaling diffusion transformer for generalist vision-language-action policy. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 7686–7697. IEEE, 2025.

Michael F Hutchinson. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 18(3):1059–1076, 1989.

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.

Miracle Kang, Lights Shi, Lucy Liang, Roy Gan, Dongxiu Liu, Pushi Zhang, Sylas Chen, Shawn Qin, Yinan Zheng, Jinliang Zheng, et al. X-tokenizer: A multimodal action tokenizer for vision-language-action pretraining. *arXiv preprint arXiv:2606.14752*, 2026.

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, 2024. *URL https://arxiv. org/abs/2406.09246*, 1(2):4, 2024.

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

David Klindt, Yann LeCun, and Randall Balestriero. When does lejepa learn a world model? *arXiv preprint arXiv:2605.26379*, 2026.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, 2009.

Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024.

Xinghang Li, Peiyan Li, Long Qian, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Xinlong Wang, Di Guo, et al. What matters in building vision–language–action models for generalist robots. *Nature Machine Intelligence*, 8(2):158–172, 2026.

Yunfei Luo, Xi Chen, Yuliang Chen, Lanshuang Zhang, Md Mofijul Islam, Siwei Zhao, Peter Kotanko, Subhasis Dasgupta, Andrew Campbell, Rakesh Malhotra, et al. Toward world modeling of physiological signals with chaos-theoretic balancing and latent dynamics. *arXiv preprint arXiv:2605.15465*, 2026.

Yueen Ma, Zixing Song, Yuzheng Zhuang, Jianye Hao, and Irwin King. A survey on vision–language–action models for embodied ai. *IEEE Transactions on Neural Networks and Learning Systems*, 2026.

Lucas Maes, Quentin Le Lidec, Damien Scieur, Yann LeCun, and Randall Balestriero. Leworldmodel: Stable end-to-end joint-embedding predictive architecture from pixels. *arXiv preprint arXiv:2603.19312*, 2026.

Oier Mees, Dibya Ghosh, Karl Pertsch, Kevin Black, Homer Rich Walke, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, et al. Octo: An open-source generalist robot policy. In *First workshop on vision-language models for navigation and manipulation at ICRA 2024*, 2024.

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. In *Robotics: Science and Systems (RSS)*, 2024.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Michael Psenka, Michael Rabbat, Aditi Krishnapriyan, Yann LeCun, and Amir Bar. Parallel stochastic gradient-based planning for world models. *arXiv preprint arXiv:2602.00475*, 2026.

Pratyaksh Rao, Wancong Zhang, Randall Balestriero, Yann LeCun, and Giuseppe Loianno. Skyjepa: Learning long-horizon world models for zero-shot sim-to-real control of quadrotors. *arXiv preprint arXiv:2606.23444*, 2026.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *European conference on computer vision*, pp. 746–760. Springer, 2012.

Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pp. 10347–10357. PMLR, 2021.

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.

Wancong Zhang, Basile Terver, Artem Zholus, Soham Chitnis, Harsh Sutaria, Mido Assran, Randall Balestriero, Amir Bar, Adrien Bardes, Yann LeCun, et al. Hierarchical planning with latent world models. *arXiv preprint arXiv:2604.03208*, 2026.

Gaoyue Zhou, Hengkai Pan, Yann LeCun, and Lerrel Pinto. Dino-wm: World models on pre-trained visual features enable zero-shot planning. *arXiv preprint arXiv:2411.04983*, 2024.

A APPENDIX

A.1 TRAINING CONFIGURATION ON ROBOCASA-GR1/ROBOTWIN 2.0

Configuration	GR1	RoboTwin 2.0
VLM backbone	Qwen3-VL-2B-Instruct	Cosmos-Reason2-2B
Action head	28-layer DiT, 1024-d	28-layer DiT, 1024-d
Training steps	100K	150K
Effective batch size	256	192
Optimizer	AdamW	AdamW
VLM learning rate	1×10^{-5}	1×10^{-5}
DiT learning rate	1×10^{-4}	1×10^{-4}
LR schedule	Warmup + cosine	Warmup + cosine
Precision	BF16	BF16
Action horizon	40	40
AdaIE (treated variant)	✓	✓
α	0.5	0.5
Sensitivity EMA	0.99	0.99
Probe interval	8 steps	8 steps
Norm matching	RMS-preserving	RMS-preserving

Table 7: Training configurations for the GR1 and RoboTwin 2.0 experiments. Within each benchmark, the baseline and AdaIE models use identical architectures, data, and optimization settings, differing only by the insertion of AdaIE at the VLM–DiT interface (bottom block).

More Details of Norm matching. To prevent sensitivity reweighting from changing the overall feature magnitude, we apply a group-wise norm-matching step after reweighting. For each group, we maintain exponential moving averages of the input and output mean-square magnitudes,

$$m_{\text{in}} \leftarrow \text{EMA}_\beta\left[\text{mean}(C^2)\right], \qquad m_{\text{out}} \leftarrow \text{EMA}_\beta\left[\text{mean}(\widetilde{C}^2)\right], \tag{6}$$

where the mean is computed over all valid tokens and feature dimensions, and $\widetilde{C} = C \odot s$ denotes the reweighted representation before normalization. We then rescale the output by

$$r = \sqrt{\frac{m_{\text{in}}}{\max(m_{\text{out}}, 10^{-12})}}, \qquad Z = r\,\widetilde{C}. \tag{7}$$

All statistics and the rescaling factor are computed without gradient tracking. This normalization preserves the relative channel-wise reweighting introduced by AdaIE while matching the overall RMS magnitude of the reweighted representation to that of the original input.

A.2 REAL-ROBOT EXPERIMENTAL DETAILS

Robot Platform. Experiments are conducted on the IRON-R01-1.11 dual-arm humanoid platform. The robot is operated as a fixed-base manipulator, with two 7-DoF arms and dexterous hands. Visual observations are captured using the head-mounted fisheye camera. The policy uses a single camera stream together with proprioceptive observations of the upper body.



Figure 5: Representative examples from the real-robot training dataset.

Dataset. The complete real-robot collection pool contains 37,362 episodes and approximately 16.8M frames collected from five R01 robots. From this pool, we construct a quality-filtered and approximately task-balanced training subset containing 4,304 episodes (13.7 hours) across 21 task categories. We additionally include 1,571 laboratory pick-and-place episodes (2.7 hours), resulting in 5,875 episodes and 16.4 hours of training data in total. Figure 5 shows representative training examples, with the data anonymized for privacy.

The original trajectories are recorded at 50 Hz and sampled at 10 Hz for training. Each sample contains a head-mounted fisheye observation, a language instruction, the current proprioceptive state, and a 30-step action chunk, corresponding to a 3-s prediction horizon. Images are cropped and resized to 320×480. We use an episode-level held-out validation split and compute normalization statistics from the training corpus.

Model and Optimization. Our real-robot policy follows a GR00T architecture. A pretrained Qwen3-VL-Instruct backbone encodes the visual observation and language instruction, while a 28-layer DiT action head is initialized from scratch and trained using flow matching. AdaIE is inserted between the VLM representation and the action DiT. It introduces no additional trainable parameters.

All baseline and AdaIE variants use identical optimization configurations. We use AdamW with BF16 training and distributed data parallelism on H200 GPUs. The pretrained VLM backbone and the from-scratch DiT action head use learning rates of 1×10^{-5} and 1×10^{-4}, respectively. For AdaIE, we set $\alpha = 0.25$. All policies are trained for approximately 32K optimizer steps with an effective batch size of 512. We compare models at matched training exposure (checkpoint at step 32K) across all variants. The complete training configuration is summarized in Table 8.

Closed-Loop Inference. At deployment, the policy predicts 30 action waypoints per inference call. The first two waypoints are discarded by the deployment controller, and five waypoints are executed before replanning. Each waypoint has a duration of 200 ms. The robot therefore repeatedly executes a short action segment, acquires a new observation, and replans in a receding-horizon fashion.

The policy outputs torso-frame end-effector targets together with joint and hand commands. These commands are passed to the robot's low-level whole-body control stack, which performs inverse-kinematics and safety-constrained execution. The perception, action-conversion, low-level control, and inference settings are held fixed across all compared variants.

Configuration	Baseline	AdaIE
VLM backbone	Qwen3-VL-2B-Instruct	Qwen3-VL-2B-Instruct
Action head	28-layer DiT, 1024-d	28-layer DiT, 1024-d
Training steps	32K	32K
Effective batch size	512	512
Optimizer	AdamW	AdamW
VLM learning rate	1×10^{-5}	1×10^{-5}
DiT learning rate	1×10^{-4}	1×10^{-4}
LR schedule	Warmup + cosine restarts	Warmup + cosine restarts
Precision	BF16	BF16
Action horizon	30	30
Flow-matching steps	4	4
AdaIE	–	✓
α	–	0.25
Sensitivity EMA	–	0.99
Probe interval	–	8 steps
Norm matching	–	RMS-preserving

Table 8: Training configurations for the real-robot baseline and AdaIE models. Both variants use identical model architectures, data, and optimization settings, differing only by the insertion of AdaIE at the VLM–DiT interface.

A.3 Training Configuration on World Models

Table 9: Training configurations for the LeWM and DINO-WM experiments. We use pre-trained backbone for two world model, with Tiny-ViT for LeWM and DINOv2 for DINO-WM.

Configuration	**LeWM**	**DINO-WM**
Predictor	6-block AdaLN-Zero DiT, 192-d	6-layer ViT, 16 heads, MLP 2048 ($\sim$19M)
Training loss	next-emb. MSE + 0.09 SIGReg	latent next-state MSE (DINOv2 space)
Proprioception	unused	excluded (pixels + actions only)
Epochs	10	10
Batch size	128	32
Optimizer	AdamW	AdamW
Encoder learning rate	1×10^{-5} (joint)†	— (frozen)
Predictor learning rate	5×10^{-5}	5×10^{-5}
Weight decay	1×10^{-3}	0
LR schedule	1% warmup + cosine	constant
Gradient clipping	1.0	1.0
Precision	BF16	BF16
Image size / frameskip	224 / 5	224 / 5
History length	3 (TwoRoom: 1)	3 (TwoRoom: 1)
AdaIE (treated variants)	✓	✓
Sensitivity EMA	0.99	0.99
Probe interval	1 step	1 step
Norm matching	RMS-preserving	RMS-preserving

A.4 Training Configuration of Section 2.2

A.4.1 Depth Estimation on NYUv2

Table 10: Training configuration for the DINOv2-S + DPT experiments on NYUv2, following the official Depth Anything V2 metric-depth recipe (SiLog loss, AdamW, polynomial decay, 518×518 crops) with the `train.py` default of 40 epochs and a global batch of 16. The baseline and AdaIE models are identical except for the insertion of AdaIE at the encoder–decoder interface (bottom block). The standard deviation reflects variation across 3 random seed training in Table 1.

Configuration	**DINOv2-S + DPT / NYUv2**
Task / dataset	NYU Depth V2, BTS/Eigen split (24,231 train pairs, 654 test images)
Input	480×640 RGB, ImageNet mean/std
Resize / crop (train / eval)	short side to 518 (multiple of 14) + random 518×518 crop / resize only
Depth target	16-bit PNG / 1000 (m), valid range $[0.001, 10]$ m
Backbone	DINOv2 ViT-S/14, 12 blocks, width 384
Backbone init	DINOv2 self-supervised weights (`dinov2_vits14_pretrain`)
Decoder	DPT head (features 64, channels 48/96/192/384), random init
Decoder input	patch tokens from blocks $\{2, 5, 8, 11\}$ after final LayerNorm (no CLS readout)
Output	sigmoid $\times$ max depth (10 m), bilinear upsampling to 480×640 at eval
Augmentation	horizontal flip ($p = 0.5$)
Loss	scale-invariant log (SiLog), $\lambda = 0.5$
Training epochs	40
Batch size	16 (4 GPUs $\times$ 4)
Optimizer	AdamW
Learning rate (decoder)	5×10^{-5}
Backbone learning rate†	5×10^{-6}
Betas	$(0.9, 0.999)$
Weight decay	0.01
LR schedule	polynomial decay $(1 - t/T)^{0.9}$, no warmup
Gradient clipping	none
Precision	float32
EMA	none
Evaluation	654 test images every epoch, Eigen crop, depth capped at 10 m; final epoch reported
Metrics	$\delta_1, \delta_2, \delta_3$, AbsRel, RMSE, $\log_{10}$
AdaIE (treated variants)	✓
Insertion point	encoder taps $\rightarrow$ DPT decoder (input side)
Groups	4×384 (one per tap)
α	0.5/-0.5
Sensitivity EMA	0.99
Probe interval	1 step
Norm matching	RMS-preserving

A.4.2 ViT Classification on CIFAR-100

Table 11: Training configuration for the DeiT-S experiments on CIFAR-100. The schedule follows the ViT fine-tuning budget (10k steps, batch 512) with DeiT AdamW defaults. The baseline and AdaIE models are identical except for the insertion of AdaIE at the backbone–classifier interface. The standard deviation reflects variation across the 5 random seed training in Table 1.

Configuration	**DeiT-S / CIFAR-100**
Task / dataset	CIFAR-100 (49k train, 1k minival held out from train, 10k test)
Input	32×32 RGB, bicubic upsampling to 224×224, ImageNet mean/std
Backbone	DeiT-S (`deit_small_patch16_224`, timm), 12 blocks, width 384, patch 16
Backbone init	ImageNet-1k supervised DeiT weights
Classifier head	Linear $384 \to 100$, random init (trunc. normal, std 0.02; zero bias)
Head input	CLS token after final LayerNorm (pre-logits)
Augmentation (train / eval)	random crop (4-px pad) + horizontal flip at 32×32 / none
Mixup / CutMix / RandAugment	none
Label smoothing / dropout	0 / 0
Stochastic depth	0.1
Training steps	10,000 ($\approx$105 epochs)
Batch size	512
Optimizer	AdamW
Learning rate (head)	1×10^{-3}
Backbone learning rate†	1×10^{-4}
Betas	$(0.9, 0.999)$
Weight decay	0.05 (none on biases, norms, pos. embed, CLS token)
LR schedule	cosine to 1×10^{-6}, 500 warmup steps
Gradient clipping	global norm 1.0
Precision	bfloat16 autocast
EMA	none
AdaIE (treated variants)	✓
Insertion point	pre-logits $\to$ classifier head
α	0.5/-0.5
Sensitivity EMA	0.99
Probe interval	1 step
Norm matching	RMS-preserving

A.4.3 DIFFUSION POLICY ON PUSH-T

Table 12: Training configuration for the DiffusionPolicy-C experiments on Push-T, following the official Diffusion Policy recipe. The baseline and AdaIE use identical configurations except for the insertion of AdaIE at the observation–denoiser interface. Following the original evaluation protocol, we report the best checkpoint performance for each variant as the mean and standard deviation of test scores across three random seeds.

Configuration	**DiffusionPolicy-C**
Task / dataset	PushT image (`pusht_cchi_v7_replay`, 90 training episodes, 2% val)
Observation	96×96 RGB + 2-D agent position, $T_o = 2$
Vision encoder	Pre-trained ResNet-18, GroupNorm, spatial-softmax pooling
Random crop (train / eval)	84×84 / fixed
Denoiser	1-D convolutional UNet (512, 1024, 2048)
Conditioning	FiLM (scale + shift), diffusion-step embed 128
Diffusion	DDPM, 100 train / 100 inference steps, squared-cosine β, ϵ-prediction
Action horizon / executed	16 / 8
Training epochs	3050
Batch size	64
Optimizer	AdamW
Learning rate	1×10^{-4}
Betas	$(0.95, 0.999)$
Weight decay	1×10^{-6}
Backbone learning rate†	1×10^{-5}
LR schedule	cosine, 500 warmup steps
EMA	power 0.75, max decay 0.9999
Evaluation	50 episodes every 50 epochs (max 300 steps);
AdaIE (treated variants)	✓
Insertion point	encoder features $\rightarrow$ denoiser conditioning (input side)
α	0.25/-0.25
Sensitivity EMA	0.99
Probe interval	1 step
Norm matching	RMS-preserving

A.4.4 VLM–DIT ON ROBOCASA-GR1

All compared variants use the same optimization configuration. We train with AdamW using $\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 10^{-5}, and gradient clipping at 1.0. The learning rate follows a cosine schedule with 5% warmup. We use a learning rate of 10^{-4} for the action DiT and 10^{-5} for the VLM backbone. We trained 30K steps for all variants in this experiment.

For sensitivity estimation, we use one Hutchinson-style probe every 8 optimization steps and an EMA decay of 0.99. For the controlled experiment in Section 2.2, the three variants differ only in the exponent α: with $\alpha = +\frac{1}{2}$ for *Equalize*, $\alpha = 0$ for the *Baseline*, and $\alpha = -\frac{1}{2}$ for *Exploit*.

Tasks. The diagnostic study uses 12 RoboCasa GR1-tabletop tasks, with 1,000 demonstrations per task. The set contains 6 articulated-receptacle pick-and-place tasks, where the policy places an object into a cabinet, drawer, or microwave and then closes the receptacle: `PnPBottleToCabinetClose`, `PnPCanToDrawerClose`, `PnPCupToDrawerClose`, `PnPMilkToMicrowaveClose`, `PnPPotatoToMicrowaveClose`, and `PnPWineToCabinetClose`.
The remaining 6 tasks are novel-object surface-to-receptacle transfers from the SplitA subset: Cuttingboard→Basket, Cuttingboard→Pot, Placemat→Bowl, Placemat→Tieredshelf, Plate→Pan, and Tray→Cardboardbox. We use 60 rollouts per task in the evaluation. The standard deviation reflects variation across the 5 random seed training in Table 1.

Table 13: Per-task success rates on RoboCasa-GR1 for our GR00T baselines and AdaIE. 60 rollouts per task.

Task	GR00T-2B	GR00T-2B + AdaIE	GR00T-4B	GR00T-4B + AdaIE
PnPBottleToCabinetClose	50.00	**56.67**	**65.00**	48.33
PnPCanToDrawerClose	65.00	**78.33**	**80.00**	75.00
PnPCupToDrawerClose	**23.33**	**23.33**	**23.33**	15.00
PnPMilkToMicrowaveClose	**53.33**	51.67	**60.00**	**60.00**
PnPPotatoToMicrowaveClose	26.67	**33.33**	41.67	**43.33**
PnPWineToCabinetClose	**48.33**	**48.33**	**51.67**	**51.67**
PnPNovelFromCuttingboardToBasket	**46.67**	45.00	56.67	**61.67**
PnPNovelFromCuttingboardToCardboardbox	38.33	**51.67**	33.33	**41.67**
PnPNovelFromCuttingboardToPan	56.67	**58.33**	**56.67**	55.00
PnPNovelFromCuttingboardToPot	45.00	**53.33**	**56.67**	53.33
PnPNovelFromCuttingboardToTieredbasket	40.00	**46.67**	43.33	**45.00**
PnPNovelFromPlacematToBasket	40.00	**45.00**	46.67	**65.00**
PnPNovelFromPlacematToBowl	46.67	**58.33**	41.67	**55.00**
PnPNovelFromPlacematToPlate	45.00	**56.67**	50.00	**63.33**
PnPNovelFromPlacematToTieredshelf	**26.67**	11.67	**31.67**	26.67
PnPNovelFromPlateToBowl	**45.00**	**45.00**	61.67	**65.00**
PnPNovelFromPlateToCardboardbox	33.33	**41.67**	36.67	**40.00**
PnPNovelFromPlateToPan	31.67	**60.00**	**43.33**	**43.33**
PnPNovelFromPlateToPlate	43.33	**63.33**	55.00	**61.67**
PnPNovelFromTrayToCardboardbox	43.33	**46.67**	38.33	**50.00**
PnPNovelFromTrayToPlate	53.33	**58.33**	48.33	**63.33**
PnPNovelFromTrayToPot	**55.00**	51.67	53.33	**58.33**
PnPNovelFromTrayToTieredbasket	**45.00**	40.00	**48.33**	43.33
PnPNovelFromTrayToTieredshelf	15.00	**18.33**	**28.33**	16.67
Average	42.36	**47.64**	47.99	**50.07**

A.5 Detailed Evaluation Results

Table 14: Per-task success rates on RoboTwin 2.0 for the GR00T-2B baseline and AdaIE, evaluated with 100 rollouts per task per setup. Clean uses the standard environment, while Randomized introduces variations in backgrounds, lighting, distractors, tabletop height, and language instructions. Avg denotes the mean over both setups. AdaIE improves average performance primarily under Clean. The limited gain under Randomized may reflect additional perceptual and generalization demands that are not directly addressed by interface sensitivity reweighting.

Task	**Clean**		**Randomized**		**Avg**	
	GR00T-2B	**+ AdaIE**	**GR00T-2B**	**+ AdaIE**	**GR00T-2B**	**+ AdaIE**
adjust_bottle	94.00	**99.00**	97.00	**98.00**	95.50	**98.50**
beat_block_hammer	62.00	**76.00**	**74.00**	**74.00**	68.00	**75.00**
blocks_ranking_rgb	65.00	**71.00**	**68.00**	64.00	66.50	**67.50**
blocks_ranking_size	**60.00**	47.00	**54.00**	46.00	**57.00**	46.50
click_alarmclock	**100.00**	99.00	**100.00**	**100.00**	**100.00**	99.50
click_bell	**100.00**	99.00	**100.00**	**100.00**	**100.00**	99.50
dump_bin_bigbin	90.00	**92.00**	**94.00**	92.00	**92.00**	**92.00**
grab_roller	97.00	**98.00**	**98.00**	95.00	**97.50**	96.50
handover_block	**74.00**	65.00	54.00	**61.00**	**64.00**	63.00
handover_mic	80.00	**93.00**	87.00	**92.00**	83.50	**92.50**
hanging_mug	**19.00**	4.00	**17.00**	10.00	**18.00**	7.00
lift_pot	45.00	**69.00**	45.00	**67.00**	45.00	**68.00**
move_can_pot	46.00	**48.00**	**40.00**	37.00	**43.00**	42.50
move_pillbottle_pad	63.00	**71.00**	58.00	**65.00**	60.50	**68.00**
move_playingcard_away	77.00	**94.00**	82.00	**93.00**	79.50	**93.50**
move_stapler_pad	23.00	**30.00**	27.00	**39.00**	25.00	**34.50**
open_laptop	81.00	**84.00**	**95.00**	92.00	**88.00**	**88.00**
open_microwave	36.00	**37.00**	**46.00**	43.00	**41.00**	40.00
pick_diverse_bottles	**77.00**	73.00	**67.00**	49.00	**72.00**	61.00
pick_dual_bottles	**94.00**	78.00	**88.00**	62.00	**91.00**	70.00
place_a2b_left	45.00	**56.00**	55.00	**66.00**	50.00	**61.00**
place_a2b_right	**58.00**	57.00	49.00	**58.00**	53.50	**57.50**
place_bread_basket	77.00	**83.00**	**83.00**	79.00	80.00	**81.00**
place_bread_skillet	83.00	**87.00**	**81.00**	78.00	82.00	**82.50**
place_burger_fries	82.00	**92.00**	**92.00**	89.00	87.00	**90.50**
place_can_basket	**38.00**	36.00	**42.00**	40.00	**40.00**	38.00
place_cans_plasticbox	27.00	**78.00**	33.00	**60.00**	30.00	**69.00**
place_container_plate	93.00	**94.00**	85.00	**89.00**	89.00	**91.50**
place_dual_shoes	**52.00**	51.00	57.00	**70.00**	54.50	**60.50**
place_empty_cup	95.00	**98.00**	**95.00**	86.00	**95.00**	92.00
place_fan	42.00	**49.00**	**53.00**	45.00	**47.50**	47.00
place_mouse_pad	29.00	**42.00**	31.00	**46.00**	30.00	**44.00**
place_object_basket	73.00	**84.00**	79.00	**81.00**	76.00	**82.50**
place_object_scale	64.00	**71.00**	61.00	**66.00**	62.50	**68.50**
place_object_stand	**81.00**	**81.00**	**82.00**	81.00	**81.50**	81.00
place_phone_stand	76.00	**80.00**	78.00	**80.00**	77.00	**80.00**
place_shoe	**87.00**	75.00	78.00	**79.00**	**82.50**	77.00
press_stapler	86.00	**89.00**	**89.00**	87.00	87.50	**88.00**
put_bottles_dustbin	37.00	**53.00**	**56.00**	39.00	**46.50**	46.00
put_object_cabinet	**51.00**	41.00	**41.00**	31.00	**46.00**	36.00
rotate_qrcode	76.00	**78.00**	71.00	**75.00**	73.50	**76.50**
scan_object	**76.00**	53.00	**59.00**	54.00	**67.50**	53.50
shake_bottle	**97.00**	96.00	**100.00**	96.00	**98.50**	96.00
shake_bottle_horizontally	96.00	**97.00**	**99.00**	95.00	**97.50**	96.00
stack_blocks_three	17.00	**32.00**	22.00	**29.00**	19.50	**30.50**
stack_blocks_two	56.00	**73.00**	62.00	**76.00**	59.00	**74.50**
stack_bowls_three	50.00	**55.00**	54.00	**55.00**	52.00	**55.00**
stack_bowls_two	**85.00**	78.00	**83.00**	72.00	**84.00**	75.00
stamp_seal	**55.00**	51.00	51.00	**52.00**	**53.00**	51.50
turn_switch	**54.00**	46.00	**59.00**	44.00	**56.50**	45.00
Average	66.42	**69.66**	67.42	**67.54**	66.92	**68.60**

Table 15: Ablation of the AdaIE equalization strength α on LeWM (seed 42). We report planning success rates (%) at each training epoch. *None* denotes the baseline without AdaIE. Each checkpoint is evaluated over 50 episodes using the same evaluation seed.

Task	α	**Ep. 1**	**Ep. 2**	**Ep. 3**	**Ep. 4**	**Ep. 5**	**Ep. 6**	**Ep. 7**	**Ep. 8**	**Ep. 9**	**Ep. 10**
Two Room	None	94	90	88	90	88	88	90	88	90	92
	0.05	94	94	94	90	94	94	92	94	92	90
	0.10	92	90	90	90	88	90	92	92	92	92
	0.15	94	94	92	88	92	90	90	92	94	92
	0.25	92	92	90	88	92	94	94	94	94	94
	0.35	92	90	88	88	92	94	94	92	96	94
	0.40	96	92	88	90	92	88	92	92	94	94
	0.50	96	92	92	92	94	94	94	92	94	92
Reacher	None	66	64	78	76	76	80	80	80	86	80
	0.05	62	70	78	76	66	80	82	86	76	74
	0.10	74	78	74	82	70	80	76	84	72	78
	0.15	72	66	64	76	68	86	76	80	84	82
	0.25	76	68	80	74	70	78	76	76	76	84
	0.35	72	70	66	78	72	82	84	78	80	74
	0.40	66	76	78	86	70	80	78	78	84	84
	0.50	58	78	82	76	74	86	80	78	84	84
Push-T	None	66	82	88	90	88	88	90	92	92	88
	0.05	64	86	88	86	92	90	92	92	96	96
	0.10	68	74	90	88	90	96	92	96	94	96
	0.15	80	86	86	94	94	96	100	94	98	94
	0.25	72	86	90	92	96	94	94	98	98	94
	0.35	66	84	84	92	98	94	96	94	94	92
	0.40	64	84	86	94	90	94	90	98	94	96
	0.50	62	76	84	86	94	84	90	98	90	94
OGBench-Cube	None	70	72	72	76	72	76	70	70	70	74
	0.05	78	76	72	76	80	80	78	76	76	80
	0.10	76	78	78	76	78	80	78	82	78	78
	0.15	72	68	72	74	72	76	74	72	72	72
	0.25	76	80	74	78	76	80	82	78	80	80
	0.35	80	78	82	80	82	84	80	74	78	78
	0.40	78	76	82	74	72	78	78	76	78	78
	0.50	80	76	78	78	80	78	80	76	78	78

A.6 CO-ADAPTION ON OTHER MODELS

A.6.1 DIFFUSION POLICY

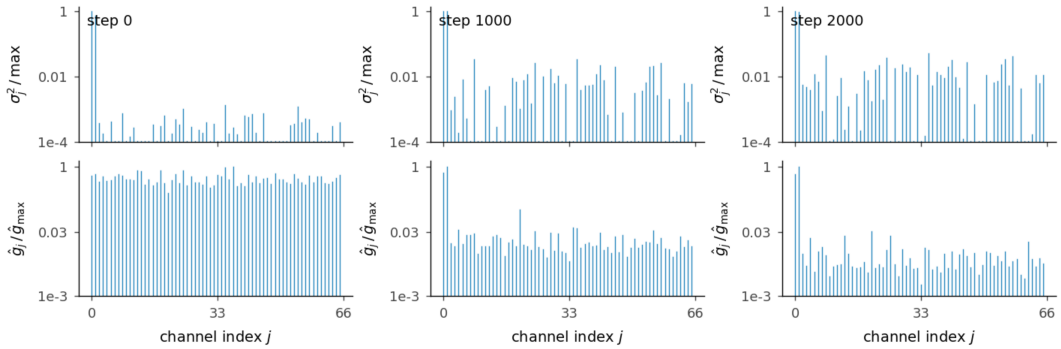


Figure 6: The co-adaption of DiffusionPolicy-C. The Policy is trained on Push-T.

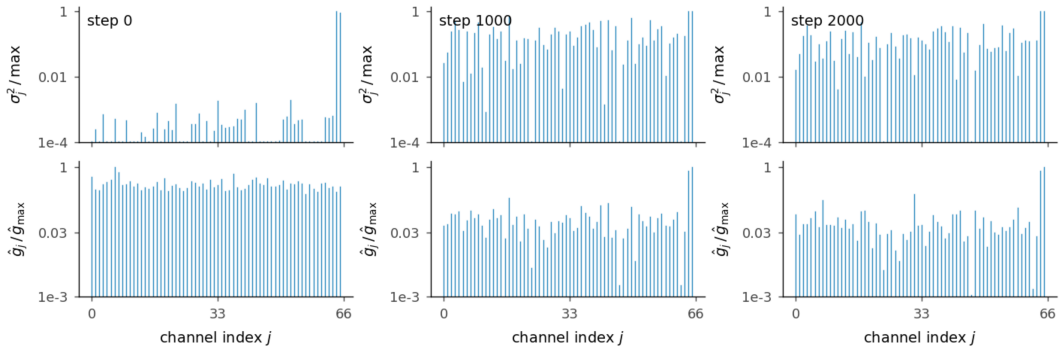


Figure 7: The co-adaption of DiffusionPolicy-T.The Policy is trained on Push-T.

A.6.2 DINO-WM

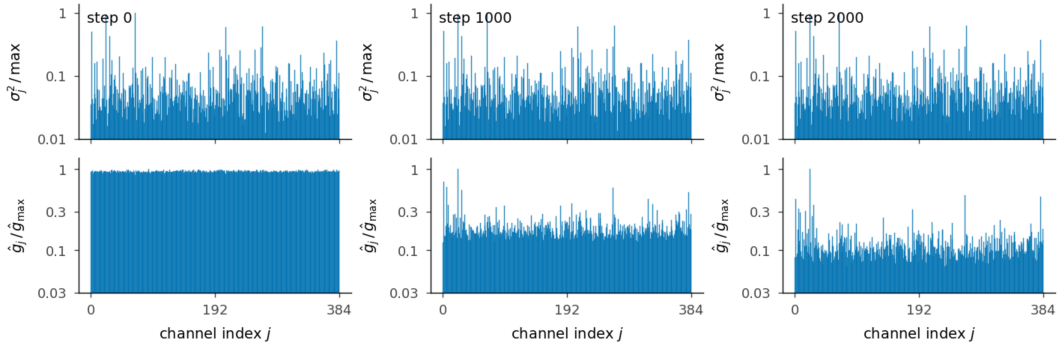


Figure 8: The co-adaption of DINO-WM.The model is trained on Two Room.