CSDR: CONTROL-STRUCTURED DISPERSIVE REGULARIZATION FOR VISION–LANGUAGE–ACTION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Vision–language–action (VLA) models transfer semantic understanding to robot control through pretrained multimodal backbones. However, representations learned through image–text pretraining may not sufficiently distinguish different control requirements. To address this issue, we propose CSDR, a control-structured representation regularization method used only during training. Its central idea is that representations need not reproduce absolute distances between numerical actions; instead, under the same instruction, samples with similar control requirements should have relatively closer representations than samples with substantially different requirements. We identify these relationships using valid action sequences from existing demonstrations and apply a ranking loss that prioritizes the refinement of representations for samples with larger action prediction errors. The corresponding ranking constraint becomes inactive once the prescribed margin is satisfied. The method introduces no teacher model, additional projection head, or inference module. Experiments with four VLA baselines on the two benchmarks, LIBERO and BridgeData V2, show that optimizing VLA representations with CSDR improves model performance. Real-world robot experiments further support this finding. These results indicate that CSDR provides a simple and transferable approach to improving VLA performance. The code is available at https://anonymous.4open.science/r/CSDR-VLA-9A7F/ and in the Supplementary Material.

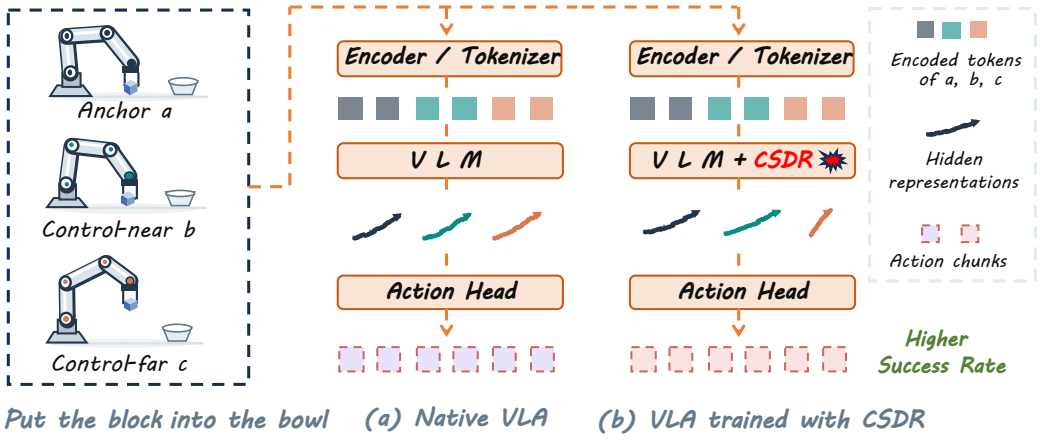


Figure 1: Overview of CSDR. Under the same instruction, action windows a and b have similar control requirements, whereas window c differs substantially from a. CSDR incorporates this relative relationship into representation learning, encouraging the representations of a and b to be closer than those of a and c. This improves the representations' ability to distinguish control requirements, supporting action prediction and task execution.

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.

1 INTRODUCTION

Embodied intelligence aims to integrate perception, language understanding, and physical interaction, enabling robots to perform diverse manipulation tasks in response to changing environments and task instructions. Rather than designing separate perception and control pipelines for individual tasks, vision–language–action (VLA) models provide a unified learning framework that directly predicts robot actions conditioned on visual observations and natural language instructions. Recent studies combining large-scale pretrained knowledge with robot demonstrations have demonstrated the potential of this framework for multitask manipulation and generalization (Brohan et al., 2023; Zitkovich et al., 2023; Kim et al., 2025b; O'Neill et al., 2024; Black et al., 2025a; NVIDIA et al., 2025; Shi et al., 2025; Huang et al., 2025).

Many VLAs use vision–language model (VLM) backbones pretrained for image–text understanding and text generation (Driess et al., 2023; Zitkovich et al., 2023; Kim et al., 2025b). However, semantic transfer does not imply that these representations adequately support action generation. Models pretrained on images and text, such as SigLIP 2, DINOv3, and Qwen, place greater emphasis on categories, semantics, and scene content (Radford et al., 2021; Tschannen et al., 2025; Siméoni et al., 2026; Bai et al., 2025b;a), whereas control also requires distinguishing motion direction, gripper state, changes in contact state, and short action sequences. For example, the instruction "put the object into the container" encompasses approaching, grasping, lifting, and placing: the same semantics can entail different control requirements.

Several studies optimize internal VLA representations through auxiliary supervision. RS-CL uses proprioceptive state distances (Kim et al., 2026), ReconVLA reconstructs manipulation-relevant regions (Song et al., 2026), FutureVLA and LIT address temporal visuomotor priors and irrelevant visual cues through latent representation alignment or pose-supervised interfaces (Xu et al., 2026; Lin et al., 2026), and Anchor-Align combines frozen-backbone anchoring with action-direction language supervision (Dalal et al., 2026). Their respective projections, reconstruction pathways, teachers, or discretized targets introduce additional training requirements. These advances raise a complementary question: can existing demonstrations directly shape the original model's hidden states to reflect control requirements without auxiliary networks?

DispLoss provides a minimally invasive precedent in generative modeling: it improves generation by dispersing hidden representations, without additional trainable modules or explicit positive pairs, and effectively improves model performance (Wang & He, 2025). However, its contrastive interpretation does not account for control relationships and is therefore not well suited to VLA models. Unconditional dispersion may push apart samples from different scenes or demonstrations even when they require similar actions. Greater dispersion does not necessarily yield better action representations; their structure should instead depend on control requirements.

Our perspective is that, under the same instruction and within a comparable control space, if windows a and b have similar control requirements while a and c differ substantially, their representations should satisfy the corresponding relative proximity. This replaces unconditional separation with reliable control relationships, without fitting absolute action distances or prescribing a global representation geometry.

Based on this perspective, we propose CSDR (Control-Structured Dispersive Regularization). The method directly reads the original model's hidden states used for action prediction and constructs control distances from valid demonstration action sequences and available proprioceptive states. For short windows at trajectory ends, it compares only shared valid real-action prefixes to prevent padded actions from distorting similarity judgments. Among same-instruction candidates, it selects mutual neighbors as reliable near neighbors and identifies far neighbors with substantially different control requirements but confusable representations. Error-weighted finite-margin ranking then corrects these relationships, prioritizing regularization for samples with larger action prediction errors.

Relational supervision also depends on having enough comparable samples. With small micro-batches imposed by memory constraints, independent random sampling may result in sparse same-instruction candidates (Chen et al., 2020). We therefore construct a cross-GPU coordinated, full-coverage *prompt cohort* plan that coordinates the instruction composition across GPUs within each forward pass and gathers features across GPUs to increase the density of comparable candidates. The plan preserves complete coverage of predefined training windows; rare samples that cannot form reliable relationships still contribute to the native task loss. CSDR thus integrates control-structure

supervision into existing VLA training without adding a teacher model, trainable projection head, or inference module.

We evaluate four baselines—OpenVLA-OFT, StarVLA, SpatialVLA, and TurboVLA (Kim et al., 2025a; StarVLA Community, 2026; Qu et al., 2025; Xie et al., 2026)—through joint training on all four LIBERO suites (Liu et al., 2023), and training on BridgeData V2 with evaluation in SimplerEnv (Walke et al., 2023; Li et al., 2025). CSDR improves these architectures without changing inference and enables some baselines to achieve state-of-the-art results under the corresponding settings. Real-robot experiments further demonstrate its transfer to practical manipulation tasks.

Our contributions are threefold:

1. We formulate VLA representation optimization as a finite-margin learning problem that learns reliable local control orderings from existing demonstrations, without fitting absolute distances.
2. We combine valid-prefix comparisons, reliable neighbor selection, and error-weighted ranking with full-coverage prompt cohorts to address relational sparsity in small batches without adding networks.
3. Across four VLA baselines, two benchmark evaluation settings, and real robots, CSDR demonstrates improved task execution while preserving the original inference architectures, adapting to diverse VLA architectures without introducing any additional network modules.

2 Related Work

2.1 Action Modeling in VLAs

VLA action modeling spans token generation, continuous generation, and regression. RT-2, OpenVLA, and FAST explore discrete action representations (Zitkovich et al., 2023; Kim et al., 2025b; Pertsch et al., 2025); Diffusion Policy, Octo, and pi0 use diffusion or flow matching (Chi et al., 2023; Ghosh et al., 2024; Black et al., 2025b). GR00T N1, DexVLA, and SmolVLA couple multimodal conditioning with action generators (NVIDIA et al., 2025; Wen et al., 2025; Shukor et al., 2025), while UniVLA and MVP-LAM learn latent actions from videos (Bu et al., 2025; Lee et al., 2026). OpenVLA-OFT combines parallel decoding, action chunking, and L1 regression (Kim et al., 2025a). SpatialVLA, StarVLA, TurboVLA, and VLA-Adapter address spatial modeling, modularity, and efficient action prediction through different architectures (Qu et al., 2025; StarVLA Community, 2026; Xie et al., 2026; Wang et al., 2025). We evaluate OpenVLA-OFT, SpatialVLA, StarVLA's Qwen-GR00T implementation, and TurboVLA to cover distinct representation structures and action-prediction mechanisms.

2.2 Research on Representation Optimization

Triplet-margin and supervised contrastive learning shape representations through sample relationships (Schroff et al., 2015; Khosla et al., 2020). In VLAs, RS-CL, ReconVLA, and Anchor-Align introduce state-based contrastive learning, visual reconstruction, or backbone anchoring with action-direction supervision (Kim et al., 2026; Song et al., 2026; Dalal et al., 2026). Their respective projections, reconstruction paths, teachers, or auxiliary targets increase training complexity. FutureVLA and LIT further explore visuomotor pretraining and pose-supervised interfaces (Xu et al., 2026; Lin et al., 2026). State or visual similarity, however, need not imply similar control requirements. In generative modeling, REPA aligns hidden states with pretrained visual features (Yu et al., 2025); REPA-E, U-REPA, and iREPA extend this approach to joint autoencoder tuning, U-Net alignment, or spatial feature structure (Leng et al., 2025; Tian et al., 2025; Singh et al., 2025). These approaches rely on external features or alignment pathways. DispLoss instead directly disperses internal representations without teachers, additional trainable parameters, or explicit positive pairs (Wang & He, 2025). CSDR adapts this minimally invasive principle to VLA control relationships, using demonstrated action windows for finite-margin ranking rather than unconditional dispersion, without introducing inference modules.

3 METHOD

3.1 PRELIMINARIES

A VLA predicts actions from images O_t, an instruction P, and an optional proprioceptive state S_t. Given a prediction horizon H and trajectory length T, overlapping windows comprise

$$X_t = (O_t, P, S_t) \tag{1}$$
$$A_t = [a_t, a_{t+1}, \ldots, a_{t+H-1}] \tag{2}$$
$$H_t = \min(H, T - t). \tag{3}$$

H_t counts real actions; terminal padding follows the baseline.

$$\mathcal{L}_{\text{task}} = \text{mean}(\mathcal{L}_{\text{native}}(X_t, A_t)) \,. \tag{4}$$

The native token cross-entropy, action regression, or diffusion/flow-matching objective remains unchanged, as do the action parameterization, prediction architecture, and inference procedure.

3.2 FULL-COVERAGE PROMPT COHORT SAMPLING

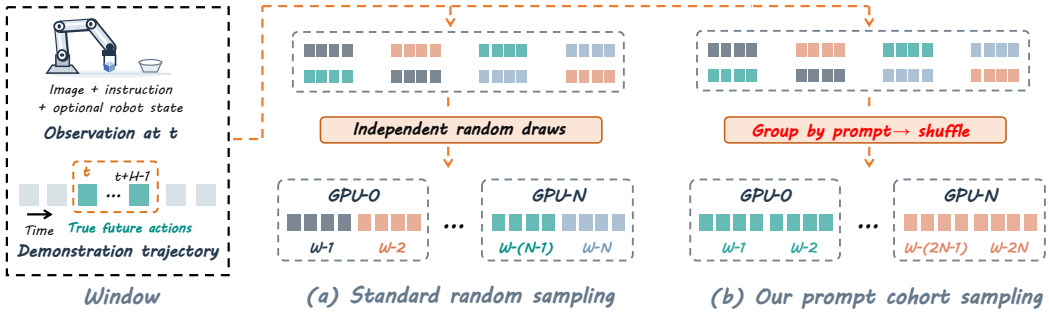


Figure 2: Full-coverage prompt cohort sampling. Windows pair current inputs with future actions. Cross-GPU instruction grouping increases same-instruction candidates within a forward pass while preserving window coverage.

As Figure 2(a) illustrates, independent sampling scatters same-instruction windows across forward passes; gradient accumulation does not enlarge the candidate set within a pass. We index trajectories, start positions, instruction IDs, and valid lengths offline. Windows are grouped by instruction, merging only conservatively verified equivalent expressions while preserving action, object, and direction distinctions. Within groups, windows are shuffled and preferably drawn from different trajectories. In Figure 2(b), each cohort coordinates R GPUs with B windows each, yielding RB same-instruction candidates; other cohorts can process different instructions. Hidden states are gathered differentiably; actions, valid lengths, and instruction IDs are gathered without gradients. Each epoch shuffles batch order while covering all predefined real windows and preserving instruction proportions. Rare or leftover windows still receive task supervision; distributed padding placeholders are masked out. Terminal windows remain eligible for CSDR when their real prefixes pass the relation filters.

3.3 CONSTRUCTING THE CSDR REGULARIZER

Demonstrated controls define supervisory orderings, whereas representation distances carry gradients; their numerical values need not match. **Control distance.** For $h = \min(H_i, H_j)$, compare translation, rotation, and gripper components q over real actions:

$$D_q(i, j; h) = \sqrt{\text{mean}\left[(A_{i,q}[:h] - A_{j,q}[:h])^2\right]}, \tag{5}$$

$$D_{\text{action}}(i, j; h) = \frac{\sum_q w_q D_q(i, j; h)/s_q(h)}{\sum_q w_q}. \tag{6}$$

The mean is taken over time and component dimensions; w_q denotes the component weights. Medians $s_q(h)$ estimated offline from training sample pairs remain fixed to avoid batch-dependent scales. Temporal differences are retained rather than averaging actions first.

$$D_{\text{control}} = \alpha D_{\text{action}} + (1-\alpha) D_{\text{state}}. \tag{7}$$

When reliable and comparable proprioceptive states are available, D_{state} uses the same component normalization, with the action term dominant. Otherwise, $D_{\text{control}} = D_{\text{action}}$, without introducing additional state inputs.

Representation and neighbors. Read Z_i from original action-query states or the fused context, without a projection head. Flattening and L2 normalization give

$$D_{\text{repr}}(i,j) = 1 - \left\langle \frac{Z_i}{\|Z_i\|_2}, \frac{Z_j}{\|Z_j\|_2} \right\rangle. \tag{8}$$

Time-aligned queries use the shared valid prefix; unified contexts are compared directly. Candidates share an instruction and a comparable control space, excluding self-pairs and invalid windows. Select up to K_{near} mutual neighbors within τ_{near}. Among candidates with control distance at least τ_{far}, select up to K_{far} with the smallest representation distances. This reduces incidental one-way adjacency and targets confusable far controls. Index selection is non-differentiable.

Common-prefix verification. For candidate triplet (i, j, k), with near j and far k, recompute both control distances over

$$h_{ijk} = \min(H_i, H_j, H_k). \tag{9}$$

$$\begin{aligned} h_{ijk} \geq H_{\min}, \quad D_{\text{control}}(i,j;h_{ijk}) \leq \tau_{\text{near}}, \\ D_{\text{control}}(i,k;h_{ijk}) \geq \tau_{\text{far}}, \\ D_{\text{control}}(i,k;h_{ijk}) - D_{\text{control}}(i,j;h_{ijk}) \geq \delta. \end{aligned} \tag{10}$$

Only triplets satisfying these conditions remain. This excludes padded-action artifacts and incomparable time spans; time-aligned representation distances also use h_{ijk}.

Finite-margin ranking. For valid triplets,

$$v_{ijk} = \max(0, D_{\text{repr}}(i,j) + m - D_{\text{repr}}(i,k)). \tag{11}$$

Satisfied triplets yield no direct ranking gradient; no absolute distance is prescribed.

$$w_i^{\text{error}} = \text{sg}\left[\text{clip}\left(\frac{e_i}{s_{\text{error}}}, 0, 1\right)\right], \tag{12}$$

$$w_{ijk}^{\text{conf}} = \text{sg}\left[\text{clip}\left(\frac{v_{ijk}}{s_{\text{conf}}}, 0, 1\right)\right], \tag{13}$$

$$w_{ijk}^{\text{length}} = \sqrt{\frac{h_{ijk}}{H}}, \tag{14}$$

$$\mathcal{L}_{\text{csdr}} = \text{mean}_{\text{valid}}\left(w_i^{\text{error}} w_{ijk}^{\text{conf}} w_{ijk}^{\text{length}} v_{ijk}\right). \tag{15}$$

Here, sg denotes stop-gradient, and $\text{clip}(x, 0, 1)$ clamps values to $[0, 1]$. For iterative generators, the native per-sample error e_i may use a training-time action reconstruction estimate without an additional complete closed-loop prediction. These weights prioritize prediction difficulty and ranking violations while downweighting short prefixes. No weight receives gradients; gradients propagate through representation distances. The mean includes only valid triplets and is zero if none exist.

Training objective.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{task}} + \lambda_t \mathcal{L}_{\text{csdr}}. \tag{16}$$

$$\lambda_t = \text{sg}\left[\min\left(1, \frac{\rho \ \text{ramp}(t)\, \mathcal{L}_{\text{task}}}{\max(\mathcal{L}_{\text{ref}}, \varepsilon)}\right)\right]. \tag{17}$$

The ramp rises from zero to one during warm-up; ρ caps the regularization-to-task-loss ratio. $\mathcal{L}_{\text{ref}}$ is the same weighted ranking mean with only w_{ijk}^{length} removed, preventing adaptive scaling from canceling short-prefix downweighting. Regularization stays active thereafter, without a final task-only phase. Only the training loss and batch organization change; no trainable parameters or inference modules are added.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Baselines and benchmarks. We evaluate OpenVLA-OFT and TurboVLA on all 40 tasks in the four LIBERO suites, and SpatialVLA and StarVLA on the four SimplerEnv WidowX tasks using their respective BridgeData V2 pipelines (Kim et al., 2025a; Xie et al., 2026; Qu et al., 2025; StarVLA Community, 2026; Liu et al., 2023; Walke et al., 2023; Li et al., 2025). LIBERO training jointly uses the Spatial, Object, Goal, and Long suites; WidowX tasks are listed in Table 1.

Training. We continue training from released task-adapted weights, preserving each baseline's action head, input processing, and task loss. CSDR adds only training-time regularization. Full-coverage epochs count predefined windows rather than trajectories; model-specific window and data settings are retained.

Evaluation. LIBERO uses 50 trials per task (2,000 per model). SpatialVLA and StarVLA each perform 24 trials per task under each of four evaluation seeds, totaling 96 trials per task.

Appendices A and B detail hardware, training configurations, and evaluation statistics.

4.2 PERFORMANCE ON TWO BENCHMARKS

We compare the closed-loop execution performance of four VLA baselines before and after applying CSDR on LIBERO and in the SimplerEnv WidowX evaluation setting with BridgeData V2 training, as shown in Table 1. The average success rates of the four baselines increase by 0.80, 0.50, 17.97, and 3.91 percentage points, respectively, indicating that CSDR's benefits extend across different architectures. The gains on LIBERO are relatively limited compared with those in the SimplerEnv WidowX evaluation setting with BridgeData V2 training. To rule out the effects of randomness, we conduct multiple independent training and evaluation runs with different random seeds, as reported in Appendix B. The results show that the performance improvements from CSDR are consistently reproducible on both benchmarks. We hypothesize that the smaller gains on LIBERO arise because the baseline models are already close to performance saturation on this benchmark, an interpretation also supported by the representation analysis in Section 4.3.

Figure 3 presents representative execution sequences in the two evaluation environments.

Table 1: Closed-loop task success rates (%) of four baselines before and after applying CSDR. The upper panel reports LIBERO results; the lower panel reports SimplerEnv WidowX evaluation results with BridgeData V2 training. Spoon, Carrot, Stack, and Eggplant denote spoon-on-towel, carrot-on-plate, green-on-yellow block stacking, and eggplant-in-basket tasks, respectively.

Method	Spatial	Object	Goal	Long	Average
OpenVLA-OFT (Kim et al., 2025a)	93.60%	99.20%	96.80%	94.60%	96.05%
OpenVLA-OFT + CSDR	**94.40%**	**99.40%**	**98.20%**	**95.40%**	**96.85%**
TurboVLA (Xie et al., 2026)	92.00%	98.60%	96.20%	94.60%	95.35%
TurboVLA + CSDR	**92.20%**	**99.00%**	**97.00%**	**95.20%**	**95.85%**
Method	Spoon	Carrot	Stack	Eggplant	Average
SpatialVLA (Qu et al., 2025)	20.83%	16.67%	20.83%	95.83%	38.54%
SpatialVLA + CSDR	**41.67%**	**37.50%**	**50.00%**	**96.88%**	**56.51%**
StarVLA (StarVLA Community, 2026)	91.67%	51.04%	43.75%	88.54%	68.75%
StarVLA + CSDR	**93.75%**	**58.33%**	**45.83%**	**92.71%**	**72.66%**

Table 2 compares the CSDR-optimized models with publicly reported results of recent VLA methods. For some methods, the code is not publicly available, so we compare against the results reported in their papers. For representation-optimization methods most closely related to our work, Appendix B.5 provides a same-backbone reproduction and comparison. OpenVLA-OFT + CSDR achieves an average success rate of 96.85% on LIBERO, while StarVLA + CSDR achieves 72.66% on SimplerEnv WidowX. These results indicate that the optimized models are competitive.

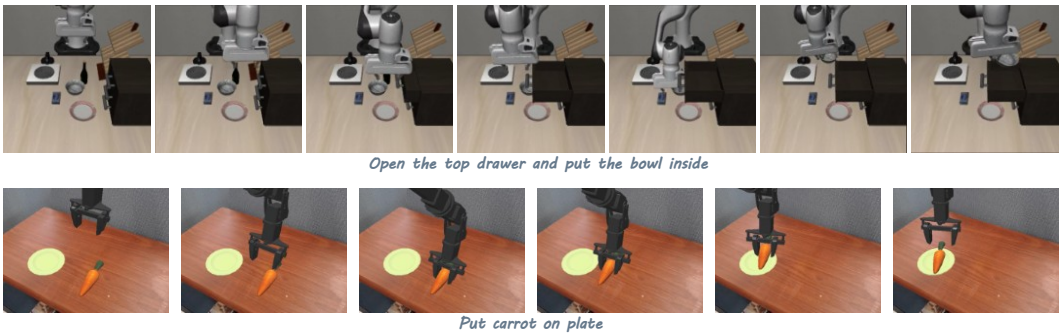


Figure 3: Representative task execution sequences in two simulated evaluation environments. The upper row shows LIBERO, and the lower row shows SimplerEnv WidowX.

Table 2: Success-rate comparison (%) between CSDR-optimized models and recent VLA methods. The upper panel reports LIBERO results, and the lower panel reports SimplerEnv WidowX results. Results for external methods are taken from their original papers. Task abbreviations follow Table 1.

Method	Spatial	Object	Goal	Long	Average
VLA-JEPA (Sun et al., 2026)	96.2%	99.6%	97.2%	95.8%	97.2%
RS-CL (Kim et al., 2026)	98.4%	98.6%	98.2%	90.4%	96.4%
OpenVLA-OFT + CSDR	**94.40%**	**99.40%**	**98.20%**	**95.40%**	**96.85%**
TurboVLA + CSDR	**92.20%**	**99.00%**	**97.00%**	**95.20%**	**95.85%**
Method	Spoon	Carrot	Stack	Eggplant	Average
VLA-JEPA (Sun et al., 2026)	75.0%	70.8%	12.5%	70.8%	57.3%
IVLR (Liu et al., 2026)	70.8%	45.8%	33.3%	91.7%	59.4%
SpatialVLA + CSDR	**41.67%**	**37.50%**	**50.00%**	**96.88%**	**56.51%**
StarVLA + CSDR	**93.75%**	**58.33%**	**45.83%**	**92.71%**	**72.66%**

4.3 Representation Optimization Analysis

The central idea of CSDR is to preserve reliable control partial orderings under the same instruction: for three demonstration windows a, b, and c, if a and b have similar control requirements while a and c differ substantially, the representations of a and b are constrained to be closer than those of a and c, improving their suitability for action generation. Section 4.2 shows that introducing CSDR improves model performance, but the association between this improvement and representation optimization requires further analysis. We therefore examine the effects of optimization at three levels.

First, we examine whether the model's internal representation geometry changes as intended. Control ordering accuracy measures the proportion of relationships in which near neighbors are closer than far neighbors in representation space. Second, we examine whether the optimized representations make action information easier to use. We fit a linear probe to predict actions; lower mean absolute error (MAE) indicates that action information is easier to read out linearly. Native-head MAE measures the discrepancy between predictions from the original action head and demonstrated actions; a lower value indicates more accurate prediction. These two metrics respectively reflect the linear readability of action information and the predictive capability of the original model, and lower values are better for both. Finally, closed-loop task success rate measures the proportion of complete tasks accomplished through continuous perception, decision-making, and execution.

We conduct paired analyses of the original weights and CSDR checkpoints for all four baselines. Each pair uses identical demonstration windows and fixed control relationships. Linear-probe training, validation, and test sets are split by trajectory to prevent neighboring windows from leaking across splits. As shown in Table 3, representation and action prediction metrics improve for all four baselines, supporting the view that the optimized representations are better suited to action generation. Meanwhile, the original representations of OpenVLA-OFT and TurboVLA already exhibit strong control discrimination on LIBERO, and their closed-loop gains after CSDR optimization are relatively limited, consistent with the analysis in Section 4.2. SpatialVLA and StarVLA show more

pronounced performance gains. Details of metric computation, probe design, and statistical procedures are provided in Appendix C.

Table 3: Representation, action prediction, and closed-loop execution metrics of four baselines before and after applying CSDR. Values are reported as "original weights → CSDR"; ↑ indicates higher is better, and ↓ indicates lower is better.

Baseline	Ordering acc. ↑	Probe MAE ↓	Head MAE ↓	Success rate ↑
OpenVLA-OFT	96.732%→ **99.738%**	0.07560→ **0.06470**	0.01773→ **0.01393**	96.05%→ **96.85%**
TurboVLA	97.931%→ **99.986%**	0.06297→ **0.04997**	0.03247→ **0.02441**	95.35%→ **95.85%**
SpatialVLA	86.652%→ **95.802%**	0.27708→ **0.24521**	0.14545→ **0.13697**	38.54%→ **56.51%**
StarVLA	85.643%→ **89.635%**	0.22585→ **0.20387**	0.11413→ **0.08281**	68.75%→ **72.66%**

4.4 Hyperparameter Analysis

We investigate three CSDR hyperparameters: the representation readout position, the regularization budget ρ, and the ranking margin m. Figure 4 shows how changes in these hyperparameters affect model performance.

When the representation readout position is varied, the maximum difference in success rate within a model does not exceed 2.34 percentage points. Adjusting ρ and m within the examined ranges also does not produce sharp performance fluctuations. These results indicate that CSDR exhibits a degree of stability within this parameter region. Further experimental details are provided in Appendix A.

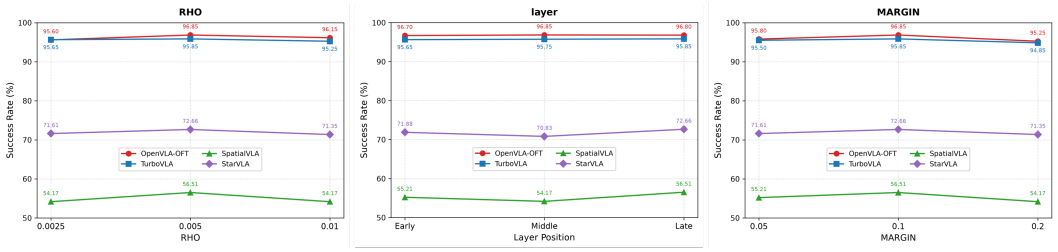


Figure 4: Sensitivity of CSDR to the representation readout position, regularization budget ρ, and ranking margin m. Early, Middle, and Late denote the relative readout positions examined within each model, rather than identical layer indices across models.

4.5 Ablation Study

To analyze CSDR's performance gains and the contributions of its components, we conduct the ablation experiments reported in Table 4.

Because all baselines continue training from their original weights, we first establish a continuation-training control using only the native task loss. Model performance changes little after continued training, demonstrating that the gains do not arise from additional training steps. As discussed in the introduction, maximizing representation dispersion to improve discriminability may disrupt the original representation structure. We apply DispLoss to VLAs, and the results show a degree of performance degradation. We then evaluate sampling-only and regularization-only configurations separately: sampling alone yields limited improvement, and regularization alone produces smaller gains than the full method. Finally, to examine the difference between CSDR's VLA-specific design and conventional triplet-margin learning, we use only a triplet-margin loss with the same sampling procedure, training budget, and feature extraction position. This yields some performance gains, but they remain smaller than those achieved by full CSDR.

These ablation experiments verify that the performance gains originate from CSDR, establish the contributions of its individual components, and demonstrate the effectiveness of its VLA-specific design. Further details of the ablation experiments are provided in Appendices B.3 and B.4.

Table 4: Ablation results (average success rate, %). OFT = OpenVLA-OFT; Turbo, Spatial, and Star abbreviate the other three baselines.

Configuration	OFT	Turbo	Spatial	Star
Task-only	96.00%	95.40%	39.58%	68.49%
DispLoss	95.95%	95.35%	38.54%	68.23%
Sampling only	96.10%	95.45%	38.54%	69.01%
Regularization only	96.50%	95.50%	44.79%	69.79%
Triplet-margin loss + sampling	96.10%	95.60%	48.18%	68.75%
Full CSDR	**96.85%**	**95.85%**	**56.51%**	**72.66%**

Table 5: Execution results on the Piper-X real robot. Gain is in percentage points (pp).

Task	Average
StarVLA baseline	81.25%
StarVLA + CSDR	**86.50%**
Gain (pp)	+5.25

4.6 Real-World Experimental Results

To validate CSDR's applicability to real robots, we construct a tabletop manipulation platform based on a Piper-X arm and use a single fixed third-person RGB camera to collect visual observations. The dataset contains eight tasks with 25 trajectories collected per task. The model is initialized from existing StarVLA Qwen2.5-VL-3B + GR00T Bridge weights and fine-tuned on the Piper demonstrations. To distinguish the effect of CSDR from conventional task learning, we use a baseline trained only with the native task loss and a comparison model with CSDR. Both share the same initialization, data splits, training budget, and checkpoint selection rules. After training, each model is evaluated for 100 trials per task, keeping the execution frequency, executed action-chunk length, maximum task duration, and control constraints consistent, while adjusting the initial layout for each trial. As shown in Table 5, CSDR also outperforms the baseline in real-robot experiments. Representative execution sequences appear in Figure 5; Appendix D provides details of dataset construction, model training, and real-robot evaluation.

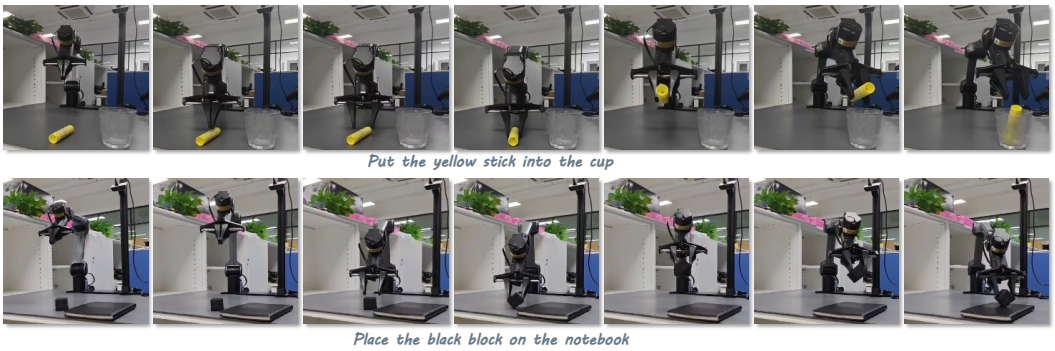


Figure 5: Representative execution sequences on the Piper-X real robot. The upper row shows "put the yellow stick into the cup," and the lower row shows "put the black block onto the notebook." Each sequence proceeds from left to right in time.

5 Conclusion

We propose CSDR, a training-time regularizer that optimizes VLA hidden representations using existing control demonstrations. It encodes reliable control partial orderings among same-instruction windows through valid-action-prefix comparisons, near/far-neighbor selection, and error-weighted finite-margin ranking. Full-coverage prompt cohort sampling increases cross-GPU relational candidate density. CSDR requires no additional teachers, trainable projection heads, or inference modules. It improves average task success rates on LIBERO, in SimplerEnv with BridgeData V2 training, and in Piper-X real-robot experiments. Representation analysis provides associated evidence of improved control ordering and action readability; ablations support the complementary roles of same-instruction sampling and representation regularization; and hyperparameter analysis indicates some stability within the examined ranges. These results suggest that relative control relationships offer a viable way to improve VLA representations while preserving the inference architecture.

AI USE STATEMENT

In this study, we used generative AI tools to assist in verifying the training and evaluation code for the method and polishing the language of the final manuscript. The authors verified the AI-assisted content, including testing the code implementation and checking the revised wording. The authors take full responsibility for the method descriptions, experimental results, conclusions, and publicly released code in the final submission.

REPRODUCIBILITY STATEMENT

To support the reproducibility of this study, we describe CSDR's relation construction, same-instruction cohort sampling, and regularization objective in the main text, and provide each baseline's training configuration, checkpoint selection procedure, evaluation protocol, and representation diagnosis details in the appendices. The baseline models and public datasets used in the experiments are specified in the paper. The training, evaluation, and inference code is publicly available in an anonymous repository (https://anonymous.4open.science/r/CSDR-VLA-9A7F/), which provides environment dependencies, download links for the initial model weights, and instructions for running the code to support reproducing CSDR continuation training and performance evaluation on the corresponding baselines.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, et al. Qwen3-VL Technical Report. *arXiv preprint arXiv:2511.21631*, 2025a. URL https://arxiv.org/abs/2511.21631.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, et al. Qwen2.5-VL Technical Report. *arXiv preprint arXiv:2502.13923*, 2025b. URL https://arxiv.org/abs/2502.13923.

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, et al. $\pi_{0.5}$: a Vision-Language-Action Model with Open-World Generalization. In *Proceedings of The 9th Conference on Robot Learning*, volume 305, pp. 17–40, 2025a. URL https://proceedings.mlr.press/v305/black25a.html.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, 2025b. doi: 10.15607/RSS.2025.XXI.010. URL https://www.roboticsproceedings.org/rss21/p010.html.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, et al. RT-1: Robotics Transformer for Real-World Control at Scale. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.025. URL https://www.roboticsproceedings.org/rss19/p025.html.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, et al. Learning to Act Anywhere with Task-centric Latent Actions. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.014. URL https://www.roboticsproceedings.org/rss21/p014.html.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A Simple Framework for Contrastive Learning of Visual Representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119, pp. 1597–1607, 2020. URL https://proceedings.mlr.press/v119/chen20j.html.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, et al. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, 2023. doi: 10.15607/RSS.2023.XIX.026. URL https://www.roboticsproceedings.org/rss19/p026.html.

Dwip Dalal, Shivansh Patel, Chahit Jain, Jeonghwan Kim, Utkarsh Mishra, Alex Baratian, et al. Generalizable VLA Finetuning via Representation Anchoring and Language-Action Alignment. *arXiv preprint arXiv:2607.13429*, 2026. URL https://arxiv.org/abs/2607.13429.

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, et al. PaLM-E: An Embodied Multimodal Language Model. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202, pp. 8469–8488, 2023. URL https://proceedings.mlr.press/v202/driess23a.html.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, et al. Octo: An Open-Source Generalist Robot Policy. In *Proceedings of Robotics: Science and Systems*, 2024. doi: 10.15607/RSS.2024.XX.090. URL https://www.roboticsproceedings.org/rss20/p090.html.

Chi-Pin Huang, Yueh-Hua Wu, Min-Hung Chen, Frank Wang, and Fu-En Yang. ThinkAct: Vision-Language-Action Reasoning via Reinforced Visual Latent Planning. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 82782–82802, 2025. doi: 10.52202/085713-2773. URL https://papers.neurips.cc/paper_files/paper/2025/hash/7749f9c0d5ff109231be21e910a3ced2-Abstract-Conference.html.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, et al. Supervised Contrastive Learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 18661–18673, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/d89a66c7c80a29b1bdbab0f2a1a94af8-Abstract.html.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*, 2025a. doi: 10.15607/RSS.2025.XXI.017. URL https://www.roboticsproceedings.org/rss21/p017.html.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, et al. OpenVLA: An Open-Source Vision-Language-Action Model. In *Proceedings of The 8th Conference on Robot Learning*, volume 270, pp. 2679–2713, 2025b. URL https://proceedings.mlr.press/v270/kim25c.html.

Taeyoung Kim, Jimin Lee, Myungkyu Koo, Dongyoung Kim, Kyungmin Lee, Changyeon Kim, et al. Contrastive Representation Regularization for Vision-Language-Action Models. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026. URL https://icml.cc/virtual/2026/poster/62819.

Jung Min Lee, Dohyeok Lee, Seokhun Ju, Taehyun Cho, Jin Woo Koo, Li Zhao, et al. MVP-LAM: Learning Action-Centric Latent Action via Cross-Viewpoint Reconstruction. *arXiv preprint arXiv:2602.03668*, 2026. URL https://arxiv.org/abs/2602.03668.

Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. REPA-E: Unlocking VAE for End-to-End Tuning of Latent Diffusion Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 18262–18272, 2025. URL https://openaccess.thecvf.com/content/ICCV2025/html/Leng_REPA-E_Unlocking_VAE_for_End-to-End_Tuning_of_Latent_Diffusion_Transformers_ICCV_2025_paper.html.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Oier Mees, Karl Pertsch, Homer Rich Walke, et al. Evaluating Real-World Robot Manipulation Policies in Simulation. In *Proceedings of The 8th Conference on Robot Learning*, volume 270, pp. 3705–3728, 2025. URL https://proceedings.mlr.press/v270/li25c.html.

Jianman Lin, Shailesh Shailesh, Zhongyi Luo, and Jiafei Duan. Breaking the Vision-Action Shortcut: Latent Interface Training for Generalizable Robotics Foundation Models. *arXiv preprint arXiv:2609.12641*, 2026. URL https://arxiv.org/abs/2609.12641.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, et al. LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791, 2023. doi: 10.52202/075280-1939. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8c3c666820ea055a77726d66fc7d447f-Abstract-Datasets_and_Benchmarks.html.

Jinkun Liu, Haohan Chi, Lingfeng Zhang, Yifan Xie, YuAn Wang, Long Chen, et al. Thinking in Text and Images: Interleaved Vision–Language Reasoning Traces for Long-Horizon Robot Manipulation. *arXiv preprint arXiv:2605.00438*, 2026. URL https://arxiv.org/abs/2605.00438.

NVIDIA, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, et al. GR00T N1: An Open Foundation Model for Generalist Humanoid Robots. *arXiv preprint arXiv:2503.14734*, 2025. URL https://arxiv.org/abs/2503.14734.

Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, et al. Open X-Embodiment: Robotic Learning Datasets and RT-X Models : Open X-Embodiment Collaboration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6892–6903, 2024. doi: 10.1109/icra57147.2024.10611477. URL https://doi.org/10.1109/icra57147.2024.10611477.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, et al. FAST: Efficient Action Tokenization for Vision-Language-Action Models. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.012. URL https://www.roboticsproceedings.org/rss21/p012.html.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Jiayuan Gu, et al. SpatialVLA: Exploring Spatial Representations for Visual-Language-Action Models. In *Proceedings of Robotics: Science and Systems*, 2025. doi: 10.15607/RSS.2025.XXI.011. URL https://www.roboticsproceedings.org/rss21/p011.html.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, et al. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pp. 8748–8763, 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. URL https://openaccess.thecvf.com/content_cvpr_2015/html/Schroff_FaceNet_A_Unified_2015_CVPR_paper.html.

Lucy Xiaoyang Shi, Brian Ichter, Michael Robert Equi, Liyiming Ke, Karl Pertsch, Quan Vuong, et al. Hi Robot: Open-Ended Instruction Following with Hierarchical Vision-Language-Action Models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pp. 54919–54933, 2025. URL https://proceedings.mlr.press/v267/shi25d.html.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, et al. SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics. *arXiv preprint arXiv:2506.01844*, 2025. URL https://arxiv.org/abs/2506.01844.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, et al. DINOv3. *Transactions on Machine Learning Research*, 2026. URL https://openreview.net/forum?id=2NlGyqNjns.

Jaskirat Singh, Xingjian Leng, Zongze Wu, Liang Zheng, Richard Zhang, Eli Shechtman, et al. What matters for Representation Alignment: Global Information or Spatial Structure? *arXiv preprint arXiv:2512.10794*, 2025. URL https://arxiv.org/abs/2512.10794.

Wenxuan Song, Ziyang Zhou, Han Zhao, Jiayi Chen, Pengxiang Ding, Haodong Yan, et al. ReconVLA: Reconstructive Vision-Language-Action Model as Effective Robot Perceiver. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(22):18549–18557, 2026. doi: 10.1609/aaai.v40i22.38921. URL https://ojs.aaai.org/index.php/AAAI/article/view/38921.

StarVLA Community. StarVLA: A Lego-like Codebase for Vision-Language-Action Model Developing. *arXiv preprint arXiv:2604.05014*, 2026. URL https://arxiv.org/abs/2604.05014.

Jingwen Sun, Wenyao Zhang, Zekun Qi, Shaojie Ren, Zezhi Liu, Hanxin Zhu, et al. VLA-JEPA: Enhancing Vision-Language-Action Model with Latent World Model. *arXiv preprint arXiv:2602.10098*, 2026. URL https://arxiv.org/abs/2602.10098.

Yuchuan Tian, Hanting Chen, Mengyu Zheng, Yuchen Liang, Chao Xu, and Yunhe Wang. U-REPA: Aligning Diffusion U-Nets to ViTs. In *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 11003–11024, 2025. doi: 10.52202/085713-0369. URL https://papers.neurips.cc/paper_files/paper/2025/hash/1042f681b823297054dd9f8b5fa7d95c-Abstract-Conference.html.

Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, et al. SigLIP 2: Multilingual Vision-Language Encoders with Improved Semantic Understanding, Localization, and Dense Features. *arXiv preprint arXiv:2502.14786*, 2025. URL https://arxiv.org/abs/2502.14786.

Homer Rich Walke, Kevin Black, Tony Z. Zhao, Quan Vuong, Chongyi Zheng, Philippe Hansen-Estruch, et al. BridgeData V2: A Dataset for Robot Learning at Scale. In *Proceedings of The 7th Conference on Robot Learning*, volume 229, pp. 1723–1736, 2023. URL https://proceedings.mlr.press/v229/walke23a.html.

Runqian Wang and Kaiming He. Diffuse and Disperse: Image Generation with Representation Regularization. *arXiv preprint arXiv:2506.09027*, 2025. URL https://arxiv.org/abs/2506.09027.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, et al. VLA-Adapter: An Effective Paradigm for Tiny-Scale Vision-Language-Action Model. *arXiv preprint arXiv:2509.09372*, 2025. URL https://arxiv.org/abs/2509.09372.

Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. DexVLA: Vision-Language Model with Plug-In Diffusion Expert for General Robot Control. In *Proceedings of The 9th Conference on Robot Learning*, volume 305, pp. 3094–3114, 2025. URL https://proceedings.mlr.press/v305/wen25b.html.

Hengyi Xie, Chenfei Yao, Xianjin Wu, Yingying Zhu, Dingkang Liang, Xiang Bai, et al. TurboVLA: Real-Time Vision-Language-Action Model at 32 Hz on an RTX 4090 with <1 GB VRAM. *arXiv preprint arXiv:2607.27205*, 2026. URL https://arxiv.org/abs/2607.27205.

Xiaoxu Xu, Hao Li, Jinhui Ye, Yilun Chen, Jia Zeng, Xinyi Chen, et al. FutureVLA: Joint Visuomotor Prediction for Vision-Language-Action Model. *arXiv preprint arXiv:2603.10712*, 2026. URL https://arxiv.org/abs/2603.10712.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, et al. Representation Alignment for Generation: Training Diffusion Transformers Is Easier Than You Think. In *International Conference on Learning Representations*, volume 2025, pp. 87400–87442, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/hash/d9e42b4d7163931f3689d6d6fbaa11d0-Abstract-Conference.html.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, et al. RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control. In *Proceedings of The 7th Conference on Robot Learning*, volume 229, pp. 2165–2183, 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

Appendix

A Experimental Configuration and Controls

A.1 Data, Training Windows, and Batches

Experiments use ten NVIDIA RTX A6000 GPUs (48 GB each) and two Intel Xeon Platinum 8352V CPUs. The LIBERO training data are a processed version combining all four suites: Spatial, Object, Goal, and Long contain 432, 454, 428, and 379 trajectories, respectively, totaling 1,693 trajectories covering 40 instruction-defined tasks. The trajectory count differs from the training-window count. The offline plan for the full-window version contains 261,614 windows, while the TurboVLA plan allowing real action prefixes near trajectory ends contains 273,465 starting positions.

SpatialVLA and StarVLA retain their respective Bridge RLDS and released LeRobot data-processing pipelines. Their full-coverage plans contain 813,372 and 1,893,026 windows, respectively. Because their preprocessed sample sets differ, the number of updates per epoch is not used to directly compare training efficiency. Window indices record trajectory identifiers, starting positions, instructions, and valid lengths; scale statistics are estimated using training data only.

A.2 Default CSDR Parameters and Readout Positions

By default, each anchor selects at most K_{near}=4 near neighbors and K_{far}=8 far neighbors. The near- and far-neighbor control thresholds are τ_{near}=0.8 and τ_{far}=1.0, respectively, and the minimum triplet control-distance gap is δ=0.2. Translation, rotation, and gripper component weights are 1, 0.5, and 0.25, respectively, normalized by their sum. Each component uses a fixed scale estimated offline from the corresponding training data, without varying across batches.

The defaults are m=0.1 and ρ=0.005. Regularization is introduced gradually over the first 300 optimizer updates, after which the budget is maintained. OpenVLA-OFT uses a control distance consisting of 0.8 times the action distance and 0.2 times the proprioceptive state distance; the other three implementations use action distance. Real-prefix implementations require a shared valid triplet length of at least 4 and downweight short prefixes by SQRT(H_COMMON / H).

Experiments on representation readout positions change only the layer from which CSDR extracts hidden states; native action prediction still follows the complete computation pathway. Token selection, aggregation, and regularization parameters remain unchanged. Table A1 lists the default and comparison positions for each baseline.

Table A1: Default CSDR representation readout positions for four baselines and the comparison positions used in the hyperparameter experiments.

Baseline	Default position	Comparison position one	Comparison position two
OpenVLA-OFT	Layer 24	Layer 8	Layer 32, final output
TurboVLA	Action decoder layer 3	Action decoder layer 1	Action decoder layer 2
SpatialVLA	Layer 26, final output	Layer 9	Layer 17
StarVLA	Layer 36, final output	Layer 12	Layer 24

A.3 Optimization Settings for Continued Training

All step counts in continued training refer to optimizer updates during the continuation stage; an individual forward pass within gradient accumulation is not counted as an update. The effective batch size is computed as the number of GPUs $\times$ per-GPU micro-batch size $\times$ number of gradient accumulation steps. Incomplete batches and placeholder samples at the end of the plan are handled according to the actual loss masks.

AdamW in OpenVLA-OFT, TurboVLA, and SpatialVLA uses BETAS=(0.9, 0.999) and ε=10^-8, with weight decay of 0.01, 10^-10, and 0, respectively. StarVLA uses FP32 master weights and

Table A2: Optimizers, batch sizes, and training budgets recorded for continued training.

Model and run version	Learning rate and optimizer	Per-GPU micro-batch / accumulation / effective batch	Full training budget and checkpoint schedule
OpenVLA-OFT	AdamW; 5×10^-5; constant learning rate	4 / 3 / 120; 10 GPUs	3 epochs, 2,181 steps per epoch, 6,543 total steps
TurboVLA	AdamW; peak 1×10^-5	4 / 3 / 120; 10 GPUs	3 epochs, 2,279 steps per epoch, 6,837 total steps
SpatialVLA	Fused AdamW; 5×10^-5; constant learning rate	8 / 4 / 320; 10 GPUs	1 epoch, 2,542 steps
StarVLA	Grouped AdamW: VLM peak 1×10^-6; action-head peak 1×10^-5	2 / 8 / 160; 10 GPUs	1 epoch, 11,832 steps

Adam states with BF16 model storage, and shards optimizer states using ZeroRedundancyOptimizer; BETAS=(0.9, 0.95), ε=10^-8, and weight decay is 10^-8. The learning rates of TurboVLA and StarVLA are linearly warmed up over the first 300 steps and then follow cosine decay to 10% of their peak values. This learning-rate schedule and the 300-step warm-up of the CSDR regularization budget are independent processes.

The trainable parameter scopes are as follows. OpenVLA-OFT trains LoRA on linear layers in the original backbone, with RANK=32, α=16, and DROPOUT=0, while updating the L1 action head and proprioceptive-state projector; the original backbone weights remain frozen. SpatialVLA trains LoRA on specified linear layers, with RANK=32, α=32, and DROPOUT=0. MODULES_TO_SAVE is empty in the saved adapter configuration, and no additional complete modules are included in training. TurboVLA freezes the BERT backbone while updating the text projection, DINOv3 visual encoder, multimodal interaction modules, and action prediction modules. StarVLA updates the VLM and GR00T flow action head involved in action prediction, freezing only the language output head and state encoder that are unused in this forward pathway.

A.4 Prompt Cohort Configuration and Valid-Relation Coverage

Table A3: Actual prompt cohort sizes and sampling-plan coverage in each run.

Model and run version	Groups per forward pass	GPUs per group R	Windows per GPU B	Candidates per group R×B	Real windows eligible for CSDR in the plan
OpenVLA-OFT	5	2	4	8	261,472 / 261,614, 99.946%
TurboVLA	5	2	4	8	273,312 / 273,465, 99.944%
SpatialVLA	5	2	8	16	714,576 / 813,372, 87.854%
StarVLA	1 regular same-instruction group	10	2	20	1,569,180 / 1,893,026, 82.893%

The candidate counts in Table A3 indicate the capacity of a regular same-instruction group before excluding the anchor itself; they do not guarantee that every candidate passes the control thresholds. Rare instructions and the end of the plan may form mixed-instruction batches for task learning. These samples still contribute to the task loss according to the plan, but CSDR relationships are not forced. Features are first gathered across all GPUs and then filtered by instruction and validity masks. Gradient accumulation does not combine features from different forward passes into a larger relation pool.

B Benchmark Evaluation and Result Statistics

B.1 Tasks, Trial Counts, and Success Rates

Each LIBERO suite contains ten tasks with 50 trials per task, giving 500 trials per suite and 2,000 per model. The SimplerEnv WidowX tasks, in order, are putting a spoon on a towel, putting a carrot on a plate, stacking a green block on a yellow block, and putting an eggplant in a basket. SpatialVLA and StarVLA each perform 24 trials per task under each of four evaluation seeds, totaling 96 trials per task. Success follows the complete-task criterion of the corresponding environment; grasp success is not used as a substitute for complete-task success.

Table B1: Per-task successful trial counts for four baselines before and after applying CSDR. Each cell reports "successful trials / total trials," corresponding to Table 1 in the main text.

Model	Spatial / Spoon	Object / Carrot	Goal / Stack	Long / Eggplant	Total
OpenVLA-OFT	468/500	496/500	484/500	473/500	1921/2000
+ CSDR	**472/500**	**497/500**	**491/500**	**477/500**	**1937/2000**
TurboVLA	460/500	493/500	481/500	473/500	1907/2000
+ CSDR	**461/500**	**495/500**	**485/500**	**476/500**	**1917/2000**
SpatialVLA	20/96	16/96	20/96	92/96	148/384
+ CSDR	**40/96**	**36/96**	**48/96**	**93/96**	**217/384**
StarVLA	88/96	49/96	42/96	85/96	264/384
+ CSDR	**90/96**	**56/96**	**44/96**	**89/96**	**279/384**

B.2 Independent Training Repetitions and Evaluation Results for LIBERO and Bridge Baselines

Table B2 reports the evaluation results of the LIBERO and Bridge baselines after independent training repetitions.

Table B2: Evaluation results of the LIBERO baselines after independent training repetitions.

Model	Spatial / Spoon	Object / Carrot	Goal / Stack	Long / Eggplant	Total
OpenVLA-OFT	468/500	496/500	484/500	473/500	1921/2000
+ CSDR, first independent repeat	**471/500**	**497/500**	**493/500**	**476/500**	**1937/2000**
+ CSDR, second independent repeat	**471/500**	**497/500**	**490/500**	**477/500**	**1935/2000**
TurboVLA	460/500	493/500	481/500	473/500	1907/2000
+ CSDR, first independent repeat	**462/500**	**497/500**	**486/500**	**474/500**	**1919/2000**
+ CSDR, second independent repeat	**461/500**	**494/500**	**485/500**	**475/500**	**1915/2000**
SpatialVLA	20/96	16/96	20/96	92/96	148/384
+ CSDR, first independent repeat	**41/96**	**37/96**	**47/96**	**93/96**	**218/384**
+ CSDR, second independent repeat	**39/96**	**36/96**	**48/96**	**93/96**	**216/384**
StarVLA	88/96	49/96	42/96	85/96	264/384
+ CSDR, first independent repeat	**90/96**	**57/96**	**43/96**	**90/96**	**280/384**
+ CSDR, second independent repeat	**91/96**	**57/96**	**43/96**	**88/96**	**279/384**

B.3 Relation to Standard Triplet-Margin Learning and Further Validation

CSDR's ranking core uses the standard triplet-margin form. For an anchor, a control-near neighbor, and a control-far neighbor, let D_{near} and D_{far} denote the representation cosine distances from the anchor to the near and far samples. The basic violation is defined as

$$V = \max(D_{\text{near}} - D_{\text{far}} + m, 0)\,. \tag{18}$$

On an identical set of valid triplets, the standard triplet objective and the CSDR objective are respectively

$$L_{\text{triplet}} = \text{MEAN}_{\text{valid}}(V), \tag{19}$$

$$L_{\text{CSDR}} = \text{MEAN}_{\text{valid}}(W_{\text{error}} W_{\text{conf}} W_{\text{length}} V). \tag{20}$$

Here, $\text{MEAN}_{\text{valid}}$ sums over valid triplets and divides by their count rather than by the sum of their weights; when no valid triplet exists, the regularization loss is zero. W_{error} is determined by the per-sample error provided by the native action-prediction path, while W_{conf} is determined by the ranking violation; both use stop-gradient. For valid real prefixes, $W_{\text{length}} = \sqrt{H_{\text{common}}/H}$ reduces the contribution of relationships with short common prefixes; without length weighting, it is set to one. The final regularization coefficient remains controlled by the warm-up and task-loss ratio cap described in the main text.

CSDR applies the triplet-margin formulation to relationships with control semantics in VLA representations. Relation construction uses dataset-level fixed-scale normalization for demonstrated actions and proprioceptive states, selects control-near and control-far candidates under the same instruction, and rechecks the near/far relationship over the common valid real prefix. Supervision allocation then uses W_{error}, W_{conf}, and, when applicable, W_{length} to adjust the contribution of different relationships. Thus, CSDR specifies both which relationships are compared and how the finite regularization budget is allocated.

The main-text Table 4 provides initial evidence. For SpatialVLA, the average success rates of Task-only, sampling only, regularization only, triplet-margin loss with sampling, and full CSDR are 39.58%, 38.54%, 44.79%, 48.18%, and 56.51%, respectively. Full CSDR is higher than all simplified settings, indicating that combining sampling and structured regularization is more effective than using either component alone. For OpenVLA-OFT, TurboVLA, and StarVLA, full CSDR likewise obtains the highest or near-highest average success rate among the corresponding configurations.

However, the "triplet-margin loss + sampling" row is only an initial matched comparison with CSDR and cannot by itself determine the independent contributions of error weighting, violation weighting, and hard-far-neighbor selection.

To further separate these components, we conducted two paired component ablations on all four baselines, as reported in Table B3. Random eligible far neighbors preserve the candidate conditions and selection counts of full CSDR, replacing only representation-based hard-far selection with random selection. The no-error-weight variant preserves hard-far selection and all other weights, setting only $W_{\text{error}} = 1$.

Table B3: Paired component ablations of CSDR.

Configuration	OpenVLA-OFT	TurboVLA	SpatialVLA	StarVLA
Task-only	96.00%	95.40%	39.58%	68.49%
Random eligible far	96.30%	95.60%	53.13%	69.27%
No error weighting	96.60%	95.60%	52.08%	69.53%
Full CSDR	**96.85%**	**95.85%**	**56.51%**	**72.66%**

The ablation results show that CSDR's gain comes from control-structured relation supervision and candidate organization during training, rather than simply increasing representation dispersion. Random eligible far neighbors still provide a clear improvement, indicating that same-instruction control relationships and finite-margin ranking form the basic source of the gain. Hard-far selection further concentrates supervision on far relationships that are currently confused in representation space, improving the quality of the correction. Removing action-error weighting decreases the average

success rate, indicating that W_{error} primarily allocates the finite regularization budget to windows with larger action-prediction errors. CSDR retains both designs and obtains the best performance.

B.4 Action-Distance InfoNCE Comparison

The component ablations in Table B3 examine neighbor selection and action-error weighting. To further test whether the gain of full CSDR can also be obtained with a conventional contrastive objective, we additionally evaluate "action InfoNCE + sampling." This is a loss-replacement comparison and complements, rather than duplicates, the single-component ablations.

The comparison uses the same initial weights, training data and window-sampling plan, representation readout position, and training budget as CSDR. It retains the fixed-scale action distance, same-instruction filtering, mutual near-neighbor selection, and representation-confused far-neighbor selection. For each anchor–near-neighbor pair, eligible far neighbors satisfying the control-difference condition are treated as negatives, and InfoNCE increases the matching probability of the near neighbor relative to these far neighbors. The CSDR margin violation and its error and violation weights are not used. Thus, this comparison preserves the same control-relation construction and sampling organization rather than treating all other batch samples as unconditional negatives.

The results are shown in Table B4. Action InfoNCE improves performance over the task-only setting, indicating that a conventional contrastive objective can benefit from the same control-relation construction and sampling organization. Full CSDR obtains higher performance, showing the benefit of its finite-margin ranking and weighted supervision.

Table B4: Action InfoNCE and full CSDR comparison.

Configuration	OpenVLA-OFT	TurboVLA	SpatialVLA	StarVLA
Action InfoNCE + sampling	96.15%	95.50%	48.96%	69.01%
Full CSDR	**96.85%**	**95.85%**	**56.51%**	**72.66%**

B.5 Same-Backbone Comparison with RS-CL

RS-CL uses distances between robot proprioceptive states to construct soft contrastive labels, and introduces representation summarization, an auxiliary projection head, and representation augmentation to bring state supervision into VLA representation learning. CSDR also targets representation structure for action generation, but directly defines control relationships from demonstrated action windows, selects reliable near neighbors and representation-confused far neighbors within the same instruction, and allocates supervision using action-prediction error and ranking violation. This directly links the regularization target to subsequent control requirements and stops the corresponding ranking constraint once the prescribed margin is satisfied, without introducing an additional trainable network. The main-text Table 2 compares CSDR-optimized models with results reported for RS-CL. To further align the backbone, data, and update budget, we perform a direct comparison on SpatialVLA.

Because RS-CL does not release code, we implement an RS-CL-inspired SpatialVLA adaptation from its published robot-state soft-label objective and projection structure. The implementation reads hidden states from SpatialVLA's final language-model normalization layer, applies masked mean pooling over valid tokens, adds a learnable offset, and uses LayerNorm, a 2304-to-2048 linear layer, and GELU to obtain a summary representation. It then applies a two-layer 2048-to-2048-to-128 MLP projector followed by L2 normalization. For the single-view Bridge input, feature dropout with probability 0.10 forms the augmented representation.

The supervision labels are constructed from the seven-dimensional Bridge proprioceptive state using SpatialVLA's Q99 normalization. The negative state distance is converted to soft labels with a state-distance temperature of 1.0, and the contrastive temperature is 0.2. Original and augmented projected representations are gathered across GPUs with gradient propagation and optimized jointly with SpatialVLA's native action-token cross-entropy. The summary vector, adapter, and projector are trainable, as verified by nonzero gradients and parameter updates. The auxiliary term uses a

300-step warm-up and a 0.5% task-loss ratio cap; RS-CL-inspired retains the cosine-decay setting from the original paper, whereas CSDR keeps its budget after warm-up.

Both methods use the same SpatialVLA initialization, Bridge training data, and full-coverage same-instruction sampling plan. Both use 10 GPUs, a per-GPU micro-batch of 8, gradient accumulation of 4, an effective global batch size of 320, a learning rate of 0.00005, fused AdamW with zero weight decay, LoRA rank and alpha of 32, seed 42, and the same BF16, TF32, FlashAttention, and data-prefetch settings. Both comparison checkpoints are after 2,542 updates; CSDR uses the existing checkpoint-2542, while RS-CL-inspired is trained to the same number of updates.

Evaluation uses the same official SimplerEnv WidowX Bridge v0 entry point, covering the Carrot, Stack, Spoon, and Eggplant tasks. Both methods use identical scenes, initial pose configurations, action ensembling, and success criteria, with a control frequency of 5 Hz and a simulation frequency of 500 Hz. The commands were checked task by task and differ only in checkpoint and output paths. Results are shown in Table B5. Under the same backbone and update budget, CSDR obtains the higher aggregate task-completion performance. CSDR introduces no additional trainable auxiliary network, avoiding the 9,186,176 auxiliary parameters in this RS-CL adaptation, including 4,458,624 parameters in the two-layer projector. The recorded training times to the same update count are 6 h 16 min 06 s for CSDR and 6 h 55 min 03 s for RS-CL-inspired, saving 38 min 57 s. With 10 GPUs, the recorded usage decreases from 69.18 to 62.68 GPU-hours, a reduction of approximately 6.49 GPU-hours or 9.38%.

Table B5: Same-backbone comparison between SpatialVLA with RS-CL-inspired and CSDR. Each task reports successful episodes / total episodes; the average success rate is computed over the 96 evaluation episodes per task configuration. GPU-hours denote the cumulative training usage to update 2,542.

Configuration	Carrot	Stack	Spoon	Eggplant	Average	GPU-hours
SpatialVLA + RS-CL-inspired	35/96	44/96	32/96	88/96	51.82%	69.18
SpatialVLA + CSDR	**36/96**	**48/96**	**40/96**	**93/96**	**56.51%**	**62.68**

C Representation Metrics, Linear Probes, and Statistical Methods

C.1 Paired Diagnostics and Hidden-State Extraction

During diagnostics, VLA parameters are frozen and the models are switched to inference mode. Representations and native action predictions are extracted from identical windows for the original and CSDR weights. Geometric relationships are constructed in advance from demonstrated actions, fixed scales, and instructions, rather than reselected according to the representations of the models being compared, preventing different models from being evaluated on relation sets of different difficulty. The diagnostic set is split by complete trajectory; windows from the same trajectory do not cross the probe training, validation, and test splits.

For Bridge, diagnostic windows come from the official bridge-test split, which is independent of the bridge-train split used for model training. For LIBERO, diagnostic windows are expert trajectories regenerated in the LIBERO simulation environment with new environment initial states and random seeds; they are not used in VLA or CSDR policy training. Thus, probe test trajectories are disjoint from probe-training trajectories and are also outside the policy-training data. These results measure the control structure and action readability of CSDR-optimized frozen representations on unseen trajectories.

The diagnostic experiments use a fixed random seed of 20260917. LIBERO covers 40 exact instructions. For Bridge, 60 exact-instruction groups are selected from those containing at least ten eligible trajectories, with approximately five positions sampled per trajectory.

For the offline SpatialVLA diagnostics, only the image and language instruction are passed to the native action-generation interface; ground-truth action tokens, action suffixes, and training labels are not provided as inputs. The model generates actions autoregressively, and the hidden states preceding prediction of the first 12 action tokens are concatenated for geometric analysis. The first state depends only on the image and instruction, whereas later states are conditioned on the model's own previously generated action tokens. Demonstrated actions are used only to construct control relationships and compute prediction errors; they are not fed into the diagnostic forward pass.

Table C1: Features used for representation geometry analysis and linear probing across four baselines. Geometry is computed using the original high-dimensional representations, while probe inputs undergo separate fixed-dimensional compression.

Model	Geometry analysis features	Main probe features
OpenVLA-OFT	56 action positions $\times$ 4096 dimensions	Flattened action positions
TurboVLA	12 action positions $\times$ 256 dimensions	Flattened action positions
SpatialVLA	12 pre-action-token prediction states $\times$ 2304 dimensions; includes self-generated action history and no true action prefix	Only the state preceding prediction of the first action token, with no true or self-generated action history
StarVLA	Final context token $\times$ 2048 dimensions	The same context state

C.2 Fixed Control Relationships and Geometry Metrics

Control distances are computed using the component normalization in Section 3.3. Candidates come from different trajectories sharing the same instruction and probe split. Each anchor selects up to four mutual near neighbors with control distance no greater than 0.8. Far neighbors are selected with a fixed random seed from candidates at distance at least 1.0, with up to eight far neighbors paired with each near neighbor. All three samples come from distinct trajectories and satisfy a control-distance gap of at least 0.2 over their shared valid length. A fixed set of up to 50,000 valid triplets is sampled. This diagnostic does not use model-specific hard-far-neighbor mining results.

$$D_{\text{near}} = 1 - \frac{\langle Z_A, Z_B \rangle}{\|Z_A\|_2 \|Z_B\|_2} \tag{21}$$

$$D_{\text{far}} = 1 - \frac{\langle Z_A, Z_C \rangle}{\|Z_A\|_2 \|Z_C\|_2} \tag{22}$$

$$g = D_{\text{far}} - D_{\text{near}} \tag{23}$$

$$\text{Acc}_{\text{order}} = \text{mean}(\mathbf{1}[g > 0]) \times 100\% \tag{24}$$

$\text{Acc}_{\text{order}}$ is the control ordering accuracy: the proportion of relationships in which the control-near sample is closer than the control-far sample in representation space. It is not equivalent to closed-loop task success rate. We first compute the proportion within each valid instruction group and then macro-average across instructions, preventing instructions with more relationships from dominating the results.

C.3 Linear Probes and Native-Head Errors

For the two sets of weights of each model, we separately fit linear probes with identical structures while keeping policy parameters frozen. We first compress hidden states using a fixed, action-label-independent 512-dimensional CountSketch, and then fit ridge regression with an intercept. The means and variances used for feature standardization are estimated only from the probe training set. The regularization coefficient is selected from 0.01, 0.1, 1, and 10 according to validation error; the test set is not used for tuning. The original and CSDR versions of each model use the same compression mapping, sample splits, and candidate parameters.

$$\widehat{A}_{\text{probe}} = WF + b \tag{25}$$

$$\text{MAE}_{\text{probe}} = \text{mean}\left(\left|\widehat{A}_{\text{probe}} - A_{\text{demo}}\right|\right) \tag{26}$$

$$\text{MAE}_{\text{native}} = \text{mean}\left(\left|\widehat{A}_{\text{native}} - A_{\text{demo}}\right|\right) \tag{27}$$

Here, F is the probe feature, W and b are its fitted weight and intercept, A_{demo} denotes demonstrated actions, and $\widehat{A}_{\text{probe}}$ and $\widehat{A}_{\text{native}}$ denote probe and native-head predictions, respectively. The main metric compares normalized actions over the first four future steps, with seven dimensions per step, yielding 28 predicted values per window. MAE is averaged over valid temporal positions and action dimensions; padded actions are excluded. Native-head MAE uses the model's own predictions, whereas linear-probe MAE uses an additionally fitted diagnostic readout, with lower values preferred for both. The probe is used only for offline analysis and is not added to the actual policy's training objective or deployment architecture.

C.4 Task-only Continuation Representation Diagnostics

To verify that the representation optimization truly comes from CSDR, we also analyze models after task-only continuation. As shown in Table C2, the representations change little after task-only continuation, supporting the conclusion that the internal representation optimization is caused by CSDR.

Table C2: Representation, action prediction, and closed-loop metrics of four baselines before and after task-only continuation. Values are reported as "original weights → task-only continuation"; ↑ indicates higher is better and ↓ indicates lower is better.

Baseline	Control ordering accuracy ↑	Linear-probe MAE ↓	Native-head MAE ↓	Closed-loop success ↑
OpenVLA-OFT	96.732%→96.943%	0.07560→0.07551	0.01773→0.01794	96.05%→96.00%
TurboVLA	97.931%→97.900%	0.06297→0.06276	0.03247→0.03237	95.35%→95.40%
SpatialVLA	86.652%→86.775%	0.27708→0.27413	0.14545→0.14507	38.54%→39.58%
StarVLA	85.643%→85.479%	0.22585→0.22183	0.11413→0.11474	68.75%→68.49%

D Real-Robot Data and Evaluation Details

D.1 Platform and Demonstration Organization

The real-robot platform consists of a Piper-X arm, a gripper, and one fixed third-person Dabai RGB camera, with an image resolution of 1280×720. The camera covers the manipulation region and target placement locations; no additional views are constructed by duplicating the single image stream.

The experiments include 8 object-placement tasks involving objects such as a yellow stick, a gray-white eraser, a black block, and a yellow clip, with a cup or a notebook serving as the target. All 8 tasks are shown in Figure D1.

D.2 Demonstration Collection, Temporal Alignment, Action Definition, and Model Training

We collect 25 demonstrations per task, totaling 200 trajectories. Each trajectory records the task instruction, images, end-effector pose, gripper state, and timestamps. The target camera recording frequency is 20 Hz, and data are exported at a target rate of 5 Hz, corresponding to a target interval of 0.2 seconds. The nearest real observation is selected at each temporal grid point. A window consists of the current image and instruction together with up to 16 future actions; at 5 Hz, a complete window nominally spans approximately 3.2 seconds. For terminal windows with fewer than 16 steps, the true length is retained and the final real action is repeated for tensor padding. In both configurations, the native flow task loss is computed over the padded 16 steps. CSDR compares only the shared real prefix and revalidates the near/far relationships at the length shared by the triplet. This shared length must be at least 4; windows shorter than 4 steps still contribute to task learning.

The first six action dimensions are linearly normalized using q01/q99 from the Piper training set and clipped to [-1,1], while the gripper dimension remains in [0,1]. Constant dimensions are normalized to 0 and mapped back to their constant values during denormalization. The initialization model is StarVLA Qwen2.5-VL-3B + GR00T Bridge. At the start of Piper fine-tuning, only model parameters are loaded, and the optimizer is reinitialized. Task-only and Task+CSDR use the same initialization, data splits, image processing, training budget, and fixed sampling plan, differing only in whether CSDR is enabled. A cohort uses R=10 GPUs with B=2 windows per GPU, providing up to 20 same-instruction candidates within a forward pass; gradient accumulation over 4 steps yields an effective task batch size of 80. CSDR reads the 2048-dimensional hidden state of the final valid context token at the last VLM layer, without inputting ground-truth action tokens. K_{near}=4, K_{far}=8, m=0.1, s_{error}=0.5, and s_{conf}=0.1. The near-neighbor control threshold is 0.8, the far-neighbor threshold is 1.0, and the minimum control-distance gap is 0.2. The weights for translation, rotation, and gripper components are 1/0.5/0.25.

D.3 Closed-Loop Execution and Runtime

The server performs StarVLA inference on a single GPU in BF16 precision, and inference results are transmitted to the robot side through SSH port forwarding. At each iteration, one image and the text instruction are sent to predict H=16 action steps. Only K_EXEC=1 step is executed before observing again. The model does not receive state inputs, but execution uses the robot's current feedback pose.

The maximum wall-clock duration is 60 seconds, and the maximum number of execution steps is 300. Timing starts when the robot connection is complete and the trial loop begins, including the first inference and network waiting time but excluding model loading and robot connection. The success and failure criteria are provided in Table D1.

Each model performs 100 trials per task, totaling 800 trials per model. Both models use the same execution frequency, number of action steps executed at a time, maximum task duration, and control constraints described above. The range of initial-layout variation is also kept consistent. Detailed execution results for each task are reported in Table D2.

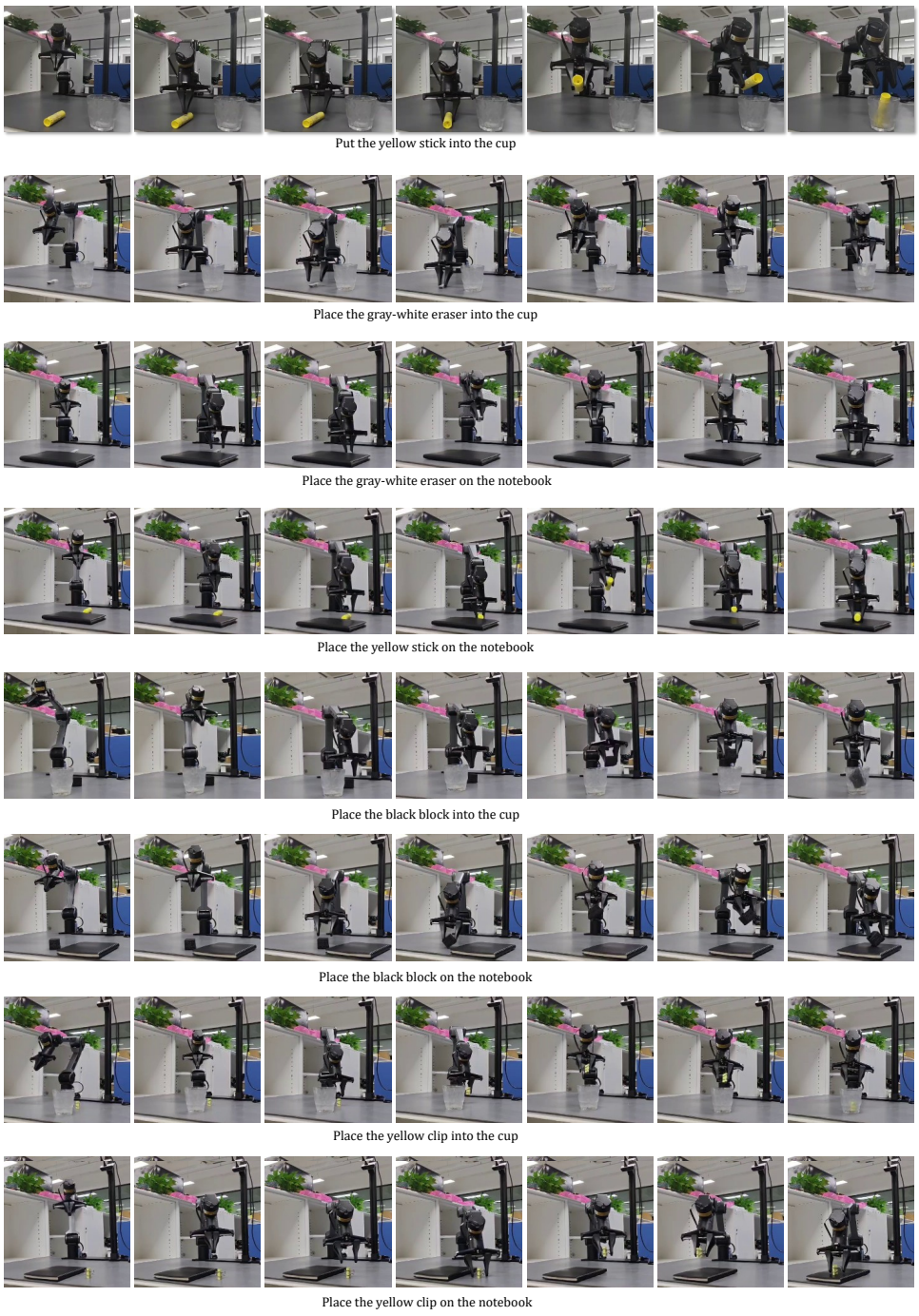


Figure D1: The eight tabletop manipulation tasks in the Piper-X real-robot experiments. Execution results for individual tasks are reported in Table D2.

Table D1: Examples of success and failure criteria.

Task category	Example of a complete success criterion
Place in the cup	The designated object enters the cup, the gripper releases it, and the object remains in the target container.
Place on the notebook	The designated object is placed in the specified region and released, and stops sliding or rolling away.
Failures and exceptions	Failure to complete before timeout, placement at an incorrect location, or dropping the object.

Table D2: Per-task execution results on the Piper-X real robot. Task numbers correspond to their order in Figure D1; cells report "successful trials / total trials (success rate)."

Task	StarVLA baseline	**StarVLA + CSDR**	Gain (percentage points)
Task 1	80/100 (80.00%)	**86/100 (86.00%)**	+6.00
Task 2	80/100 (80.00%)	**83/100 (83.00%)**	+3.00
Task 3	84/100 (84.00%)	**89/100 (89.00%)**	+5.00
Task 4	85/100 (85.00%)	**88/100 (88.00%)**	+3.00
Task 5	79/100 (79.00%)	**84/100 (84.00%)**	+5.00
Task 6	82/100 (82.00%)	**89/100 (89.00%)**	+7.00
Task 7	77/100 (77.00%)	**83/100 (83.00%)**	+6.00
Task 8	83/100 (83.00%)	**90/100 (90.00%)**	+7.00