# CONTINUAL VLA MODEL MERGING WITH HISTORICAL SUBSPACE CONSTRAINTS

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Vision–language–action (VLA) models must continually acquire new manipulation skills while preserving previously learned capabilities. Maintaining separate policies for individual skills incurs growing deployment overhead. Model merging provides a natural mechanism for consolidating skill-specific adaptations into a shared model. However, repeated adaptation and consolidation in rehearsal-free settings introduce a key challenge: current-stage supervision provides no direct signal about how incoming updates affect previously acquired skills, allowing interference to propagate and potentially accumulate across learning stages. We study rehearsal-free continual VLA model merging and propose *history-aware subspace-constrained adaptation*. Our key idea is to retain compact activation subspaces that summarize dominant directions in historical layer-input activations and use them as historical constraints during subsequent adaptation. The method constrains incoming backbone adaptation at two complementary levels: a *soft orthogonal gradient constraint* attenuates gradient components aligned with a cumulative historical subspace, while *Subspace-L2 regularization* penalizes the response induced by the effective backbone update along retained stage-specific historical input subspaces. After adaptation, the learned update is consolidated into the shared vision–language backbone, stage-specific action modules are retained, and historical subspaces are updated from stage-end rollouts, without retaining or replaying previous demonstrations. Experiments on within-suite and cross-suite LIBERO task streams show that our method achieves the highest final average success rate among the evaluated rehearsal-free approaches, reaching $45.5\%$ on the ten-task LIBERO-Goal stream and $87.6\%$ on the Spatial→Object→Goal stream.

## 1 INTRODUCTION

Vision–language–action (VLA) models Zitkovich et al. (2023); Kim et al. (2025); Black et al. (2025) leverage perceptual and semantic knowledge from vision–language pretraining to perform instruction-conditioned robotic manipulation. As robots encounter new objects, scenes, and tasks, they must integrate an expanding repertoire of manipulation skills while preserving previously acquired capabilities. Maintaining a separate full VLA policy for every skill incurs growing storage and deployment overhead, motivating approaches that consolidate skill-specific adaptations into a shared model. Model merging Wortsman et al. (2022); Ilharco et al. (2023) provides a natural mechanism by combining adapted models or their task-specific parameter updates. Recent work has applied this paradigm to robotic policies: MergeVLA Fu et al. (2026) studies cross-skill integration of independently adapted VLA policies, while RETAIN Yadav et al. (2026) shows that recursive parameter merging can support sequential skill adaptation. These studies establish the feasibility of sequential VLA model merging, but do not explicitly address how to control interference during adaptation when historical demonstrations are unavailable.

We therefore study *rehearsal-free continual VLA model merging*. Unlike one-shot merging, consolidated parameters from each stage initialize subsequent adaptation without access to previous demonstrations. Current-stage supervision therefore provides no direct signal about how incoming updates affect previously acquired skills. In the modular VLA setting considered here, a vision–language backbone is shared across learning stages, while stage-specific action modules are retained separately. As the shared backbone continues to adapt, incoming updates may perturb representations used by previously acquired skills; once consolidated, these perturbations propagate to subsequent



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.

stages and may accumulate across repeated adaptation–consolidation cycles. We therefore ask: *How can new VLA skills be learned and continually consolidated while reducing interference with previously acquired skills, without replaying historical demonstrations?*

To provide historical constraints without replay, we retain compact activation subspaces that summarize dominant directions in historical layer-input activations. These subspaces serve as references for selectively constraining incoming backbone updates. Building on this principle, we propose *history-aware subspace-constrained adaptation*, which constrains incoming backbone adaptation at two complementary levels. A *soft orthogonal gradient constraint* attenuates gradient components aligned with a cumulative historical subspace, while *Subspace-L2 regularization* penalizes activation changes induced by the backbone update in retained stage-specific historical input subspaces. After adaptation, the learned update is merged into the shared vision–language backbone, while the stage-specific action module is frozen and retained. Historical activation subspaces are updated from stage-end rollouts and carried forward to constrain subsequent adaptation without retaining previous demonstrations. We demonstrate the effectiveness of the proposed method on within-suite and cross-suite LIBERO task streams, both without replay. Our method outperforms existing rehearsal-free approaches, achieving a final success rate of 45.5% on the ten-task LIBERO-Goal stream and 87.6% on the cross-suite Spatial→Object→Goal stream. Ablation studies confirm that both constraint levels contribute to this performance.

Our contributions are threefold:

- We investigate interference in rehearsal-free continual VLA model merging, where successive adaptations may perturb representations supporting previously acquired skills, and subsequent consolidation can carry these effects into later stages.
- We propose *history-aware subspace-constrained adaptation*, which retains compact historical activation subspaces as historical constraints and controls incoming backbone updates at two complementary levels. A soft orthogonal gradient constraint attenuates gradient components aligned with a cumulative historical subspace, while Subspace-L2 regularization penalizes update-induced activation changes in retained stage-specific historical input subspaces.
- We evaluate the proposed method on both within-suite and cross-suite LIBERO task streams. Our method achieves a Final Avg. SR of $45.5\%$ on the ten-task LIBERO-Goal stream and $87.6\%$ on the Spatial→Object→Goal stream, outperforming the evaluated rehearsal-free baselines.

## 2 Related Work

### 2.1 VLA Adaptation and Model Merging

Pretrained vision–language–action (VLA) Zitkovich et al. (2023); Kim et al. (2025); Black et al. (2025) policies often require downstream adaptation to acquire manipulation skills beyond those covered by pretraining. Model merging Wortsman et al. (2022); Ilharco et al. (2023) provides a mechanism for consolidating such adaptations by combining adapted models or their parameter updates . Recent work extends this paradigm to robotic policies. MergeVLA Fu et al. (2026) studies cross-skill integration of independently adapted VLA policies and introduces a merging-oriented architecture to improve mergeability. RETAIN Yadav et al. (2026) interpolates pretrained and task-adapted policies and extends this procedure to sequential skill acquisition. These studies establish the feasibility of VLA model merging and sequential consolidation, but do not provide explicit mechanisms to prevent interference with previously acquired skills under repeated rehearsal-free adaptation. Repeated incorporation of incoming task adaptations into the consolidated policy may degrade performance on earlier skills. We mitigate this interference by constraining new adaptations to preserve historical activation subspaces.

### 2.2 Continual Model Merging

Continual model merging incrementally integrates sequentially arriving task models or parameter updates into a consolidated model. A central challenge is interference with previously integrated knowledge. Existing approaches address this challenge by manipulating incoming task vectors or

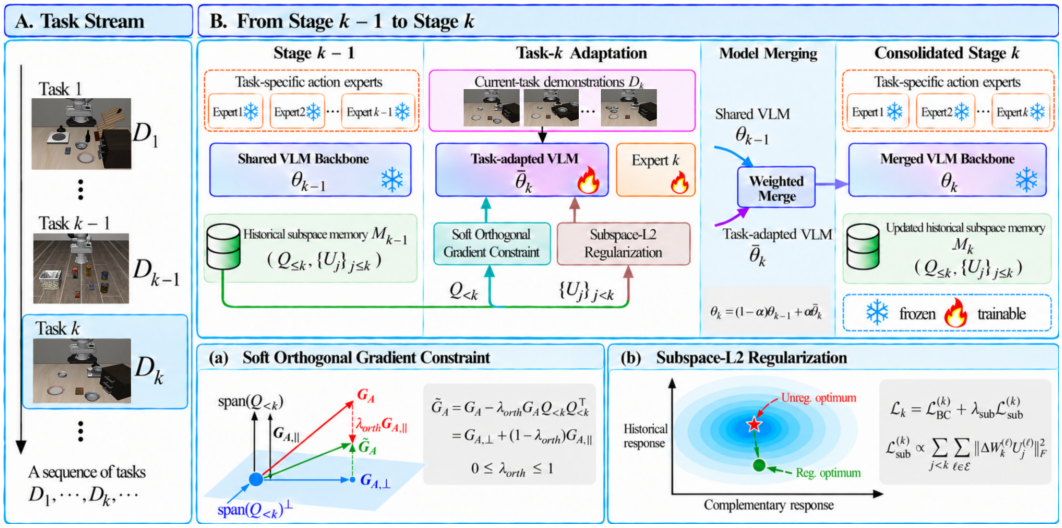


Figure 1: Overview of the proposed history-aware subspace-constrained adaptation for continual VLA model merging. At stage $k$, the consolidated backbone from stage $k-1$ is adapted on demonstrations $D_k$ under constraints from historical activation subspaces stored in $\mathcal{M}_{k-1}$. Soft Orthogonal Gradient Constraint attenuates gradients aligned with the cumulative subspace $Q_{\le k}$, while Subspace-L2 Regularization penalizes activations along stage-specific subspaces $\{U_j\}_{j \le k}$. After adaptation, the backbone update is merged, the action module is frozen, and rollout activations are compressed into $\mathcal{M}_k$ for subsequent stages.

parameter updates before or during consolidation. OPCM Tang et al. (2025) reduces overlap between incoming task vectors and accumulated updates through orthogonal projection and adaptive rescaling. NUFILT Qiu et al. (2026) links task-vector and representation subspaces and combines null-space filtering with projection-aware low-rank correction before consolidation. MagMax Marczak et al. (2025) combines sequential fine-tuning with maximum-magnitude parameter selection to consolidate task-specific updates.

In VLA policies, RETAIN Yadav et al. (2026) extends parameter merging to sequential skill acquisition by recursively merging the current adapted policy with the previously consolidated policy. RETAIN establishes the feasibility of sequential VLA model merging, but does not explicitly constrain current-stage adaptation using historical representation information. Our work addresses this gap by constraining adaptation before consolidation. In the modular VLA setting considered here, a shared vision–language backbone is repeatedly adapted across learning stages, while stage-specific action modules are frozen and retained. Incoming backbone updates may therefore perturb representations consumed by historical action modules. To control this interference without replay, we retain compact activation subspaces extracted from stage-end rollouts and use them as historical constraints during subsequent backbone adaptation.

## 3 Problem Formulation and Motivation

### 3.1 Rehearsal-Free Continual VLA Model Merging

**Continual and rehearsal-free setting.** We consider continual learning of a VLA policy over stages $k = 1, \ldots, K$. At stage $k$, a learning unit comprising one or more related manipulation tasks is accompanied by demonstrations $\mathcal{D}_k = \{\tau_i^k\}_{i=1}^{N_k}$. Each trajectory contains visual observations $o \in \mathcal{O}$, proprioceptive states $p \in \mathcal{P}$, language instructions $l \in \mathcal{L}$, and $H$-step action chunks $\mathbf{a} \in \mathcal{A}^H$. Historical demonstrations from previous stages are neither stored nor replayed during subsequent training.

**Modular VLA architecture and continual consolidation.** The policy consists of a shared vision–language backbone $f_\theta : \mathcal{O} \times \mathcal{L} \to \mathcal{Z}$ and stage-specific action modules $h_{\phi_k} : \mathcal{Z} \times \mathcal{P} \to \mathcal{A}^H$. The

stage identity selects the corresponding action module, while task semantics enter through language instructions. Let $\theta_k$ denote the consolidated backbone after stage $k$, with $\theta_0 = \theta_{\text{init}}$. During stage-$k$ adaptation, $\theta_{k-1}$ is fixed while temporary LoRA adapters Hu et al. (2022) and the incoming action module $\phi_k$ are optimized using only $\mathcal{D}_k$, producing an effective backbone update $\Delta_k$. The unscaled update is used during training, whereas the merge coefficient $\alpha > 0$ is applied only at consolidation:

$$\theta_k = \theta_{k-1} + \alpha\Delta_k. \tag{1}$$

The temporary adapters are then discarded, $\phi_k$ is frozen and retained, and $\theta_k$ initializes the next stage. After each stage, rollout activations are compressed into compact historical summaries $\mathcal{M}_k$. The retained state is therefore $\mathcal{S}_k = (\theta_k, \{\phi_j\}_{j\le k}, \mathcal{M}_k)$, with no historical demonstrations or rollout trajectories retained.

**Historical risk under a changing backbone.** For an earlier stage $j \le k$, the corresponding action module $\phi_j$ remains fixed, but its policy $h_{\phi_j}(f_{\theta_k}(o,l),p)$ changes as the shared backbone is continually updated. We define its empirical risk under behavior cloning as

$$\mathcal{R}_j(\theta,\phi_j) = \mathbb{E}_{(o,p,l,\mathbf{a})\sim\mathcal{D}_j}\left[\ell_{\text{act}}\left(h_{\phi_j}(f_\theta(o,l),p),\mathbf{a}\right)\right], \tag{2}$$

where $\mathcal{D}_j$ denotes the stage-$j$ demonstration set and $\ell_{\text{act}}$ is the action-prediction loss. Stage-$k$ adaptation minimizes the current-stage risk $\mathcal{R}_k(\theta_{k-1}+\Delta_k,\phi_k)$. If historical demonstrations were available, post-consolidation learning could additionally account for the historical risks $\{\mathcal{R}_j(\theta_{k-1}+\alpha\Delta_k,\phi_j)\}_{j\le k}$. Under rehearsal-free learning these terms are inaccessible, motivating compact historical summaries that can constrain incoming adaptation.

### 3.2 Historical Interference under Repeated Consolidation

Current-stage optimization provides no direct guarantee that consolidation preserves earlier stages. For $j < k$, let $\delta\mathcal{R}_j^{(k)} = \mathcal{R}_j(\theta_k,\phi_j) - \mathcal{R}_j(\theta_{k-1},\phi_j)$ denote the stage-$k$ induced change in stage-$j$ risk, where positive values indicate degradation. Across repeated consolidation stages, these changes accumulate as

$$\mathcal{R}_j(\theta_k,\phi_j) - \mathcal{R}_j(\theta_j,\phi_j) = \sum_{m=j+1}^{k}\delta\mathcal{R}_j^{(m)}. \tag{3}$$

Repeated positive changes therefore lead to increased historical risk, although later updates may partially offset earlier ones. Because historical risks cannot be evaluated during rehearsal-free adaptation, we seek a tractable representation-level surrogate. For a LoRA-adapted layer with effective update $\Delta W_{\ell,k} = s_\ell B_{\ell,k}A_{\ell,k}$, consolidation induces the local output perturbation $\delta z = \alpha\Delta W_{\ell,k}x$ for layer input $x$. After stage $j$, we collect layer inputs $X_{\ell,j} \in \mathbb{R}^{d_{\text{in},\ell}\times n_{\ell,j}}$ from rollout activations and retain an orthonormal basis $U_{\ell,j} \in \mathbb{R}^{d_{\text{in},\ell}\times q_{\ell,j}}$ spanning the dominant historical input-activation subspace, obtained via truncated singular value decomposition and retaining the top $q_{\ell,j}$ principal directions. Writing $x = U_{\ell,j}c + r$, with $c = U_{\ell,j}^\top x$ and $r = (I - U_{\ell,j}U_{\ell,j}^\top)x$, gives

$$|\delta z|_2 \le \alpha|\Delta W_{\ell,k}U_{\ell,j}|_F|c|_2 + \alpha|\Delta W_{\ell,k}|_2|r|_2. \tag{4}$$

When historical inputs are well represented by the retained subspace ($|r|_2$ is small), $|\Delta W_{\ell,k}U_{\ell,j}|_F$ provides a tractable surrogate for limiting local perturbations along dominant historical directions. This motivates two complementary forms of historical-subspace protection: gradient-level control of the adaptation direction and update-level regularization of the effective low-rank residual.

## 4 History-Aware Subspace-Constrained Adaptation

**Method overview.** Figure 1 summarizes the proposed method. Historical activation subspace memory contains two complementary components: stage-specific bases $\{U_j^{(\ell)}\}$ for Subspace-L2 regularization and a rank-bounded cumulative basis $Q_{<k}^{(\ell)}$ for soft orthogonal gradient control. Together, they constrain the incoming LoRA adaptation before its residual is consolidated into the shared backbone. We describe memory construction, the two constraints, and the stage-wise procedure below.

### 4.1 Historical Activation Subspace Memory

At the end of stage $k$, we execute rollouts of the consolidated stage-end policy on the current learning unit and collect layer-input activations for each adapted layer $\ell \in \mathcal{E}$. Let $X_k^{(\ell)} \in \mathbb{R}^{d_{\text{in}}^{(\ell)} \times n_k^{(\ell)}}$ contain these activation vectors as columns. We use the uncentered second-moment matrix so that the retained subspace captures dominant activation energy, including contributions from nonzero mean activations:

$$C_k^{(\ell)} = \frac{1}{n_k^{(\ell)}} X_k^{(\ell)} X_k^{(\ell)\top}. \tag{5}$$

**Stage-specific activation basis.** For each layer, we retain the smallest leading eigenspace of $C_k^{(\ell)}$ whose cumulative eigenvalue energy reaches a target fraction $\eta$, subject to a rank cap $q_{\text{hist}}$. This yields an orthonormal basis $U_k^{(\ell)} \in \mathbb{R}^{d_{\text{in}}^{(\ell)} \times q_k^{(\ell)}}$, with $q_k^{(\ell)} \le q_{\text{hist}}$. When the required rank exceeds the cap, the target energy fraction need not be fully preserved. The stage-specific bases $\{U_j^{(\ell)}\}_{j \le k}$ are retained individually for Subspace-L2 regularization.

**Cumulative historical basis.** For gradient-level control, we additionally maintain a rank-bounded basis $Q_{<k}^{(\ell)}$ that summarizes the aggregate span of previous stage-specific bases. Starting from $Q_{<1}^{(\ell)} = \emptyset$, it is updated after stage $k$ as

$$Q_{<k+1}^{(\ell)} = \text{Trunc}_{q_{\text{hist}}}\left([Q_{<k}^{(\ell)}, U_k^{(\ell)}]\right), \tag{6}$$

where $\text{Trunc}_q(Y)$ returns the leading $\min(q, \text{rank}(Y))$ left singular vectors of $Y$. Thus, the cumulative component remains rank bounded, whereas the stage-specific bases accumulate across learning stages. Raw rollout observations and activation samples are discarded after the bases are constructed.

### 4.2 Soft Orthogonal Gradient Constraint

For an adapted layer $\ell$, LoRA parameterizes the effective residual as $\Delta W_k^{(\ell)} = s^{(\ell)} B_k^{(\ell)} A_k^{(\ell)}$, where $A_k^{(\ell)} \in \mathbb{R}^{r \times d_{\text{in}}^{(\ell)}}$ and $B_k^{(\ell)} \in \mathbb{R}^{d_{\text{out}}^{(\ell)} \times r}$. We constrain the gradient of $A_k^{(\ell)}$ because it acts on the input-feature space where the historical subspace $Q_{<k}^{(\ell)}$ resides. Let $G_{A,k}^{(\ell)}$ denote its raw gradient under the stage-$k$ training objective. We attenuate the component of the gradient aligned with the historical subspace:

$$\widetilde{G}_{A,k}^{(\ell)} = G_{A,k}^{(\ell)} - \lambda_{\text{orth}} \big(G_{A,k}^{(\ell)} Q_{<k}^{(\ell)}\big) Q_{<k}^{(\ell)\top}, \tag{7}$$

where $0 \le \lambda_{\text{orth}} \le 1$. Setting $\lambda_{\text{orth}} = 0$ leaves the gradient unchanged, whereas $\lambda_{\text{orth}} = 1$ removes its component in the retained cumulative subspace. The transformation is applied only to the LoRA-$A$ gradient; gradients of $B_k^{(\ell)}$ and the trainable parameters of $\phi_k$ remain unchanged.

### 4.3 Subspace-L2 Regularization on LoRA-Induced Residuals

Soft Orth controls the optimization direction through the cumulative historical basis. We complement it with an update-level penalty using the stage-specific bases retained in memory. Motivated by the local perturbation bound in Section 3.2, we penalize the response induced by the effective residual along historical input directions. Because the merge coefficient $\alpha$ is fixed, the constant factor $\alpha^2$ can be absorbed into $\lambda_{\text{sub}}$.

For $k > 1$, we define

$$\mathcal{L}_{\text{sub}}^{(k)} = \frac{1}{k-1} \sum_{j<k} \frac{1}{|\mathcal{E}|} \sum_{\ell \in \mathcal{E}} \left\| \Delta W_k^{(\ell)} U_j^{(\ell)} \right\|_F^2. \tag{8}$$

The uniform averaging treats all retained historical stages and adapted layers equally. At $k = 1$, no historical basis is available and $\mathcal{L}_{\text{sub}}^{(1)} = 0$. Similarly, the cumulative basis $Q_{<1}^{(\ell)}$ is empty, so the soft orthogonal constraint is also inactive at the first stage. Unlike global $\ell_2$ regularization on the effective residual, this term selectively penalizes residual responses along retained historical

directions, while directions in the orthogonal complement are not directly constrained. The stage-$k$ adaptation objective is

$$\mathcal{L}_k = \mathcal{L}_{\mathrm{BC}}^{(k)} + \lambda_{\mathrm{sub}} \mathcal{L}_{\mathrm{sub}}^{(k)}, \tag{9}$$

where $\mathcal{L}_{\mathrm{BC}}^{(k)}$ is the behavioral-cloning loss on current-stage demonstrations and $\lambda_{\mathrm{sub}} \geq 0$ controls the regularization strength.

**Interaction between the two constraints.** Soft Orth is applied to the $A$-factor gradient of the joint objective, $G_{A,k}^{(\ell)} = \nabla_{A_k^{(\ell)}} \mathcal{L}_k$. Consequently, it also transforms the Subspace-L2 contribution to the $A$-factor gradient, while the $B$-factor and action-module gradients remain unmodified. The complete gradient derivation is provided in the appendix.

### 4.4 Stage-Wise Training, Consolidation, and Memory Update

At stage $k$, we (i) attach fresh LoRA adapters to the consolidated backbone $\theta_{k-1}$ and initialize the incoming action module $\phi_k$; (ii) optimize equation 9 on $\mathcal{D}_k$ while applying equation 7 to LoRA-$A$ gradients; (iii) consolidate the learned residual according to equation 1, remove the temporary adapters, and freeze $\phi_k$; (iv) collect rollout activation statistics for the current learning unit and extract $\{U_k^{(\ell)}\}_{\ell \in \mathcal{E}}$; and (v) update the cumulative bases using equation 6 and retain the updated historical memory for stage $k+1$. Complete pseudocode is provided in the appendix.

## 5 Experiments

We evaluate the proposed method with three questions in mind: **Q1: Effectiveness.** Does history-aware constrained adaptation improve continual VLA merging under repeated consolidation over long task sequences and cross-suite distribution shifts? **Q2: Components and mechanisms.** Which historical-subspace constraints contribute to the performance gains, and are their effects consistent with the intended reduction in historical interference? **Q3: Efficiency.** What additional persistent state does historical-subspace protection require, and does it alter the deployed footprint or inference-time computation? We first describe the experimental protocols, model configuration, comparison methods, and evaluation metrics (Section 5.1). We then evaluate long-sequence and cross-suite continual merging, analyze the contributions of the proposed constraints, and quantify their storage and deployment overhead.

### 5.1 Experimental Setup

**Benchmarks and protocols.** We evaluate rehearsal-free continual VLA merging in LIBERO (Liu et al., 2023) under two complementary protocols. For *within-suite long-sequence continual merging*, we construct a ten-stage LIBERO-Goal stream, with one manipulation task introduced per stage. For *cross-suite continual merging*, we use the three-stage Spatial→Object→Goal stream, where each stage introduces one complete LIBERO suite. The former stresses repeated adaptation and consolidation within a single suite, whereas the latter introduces consecutive cross-suite distribution shifts. At each stage, adaptation uses only current-stage demonstrations, and the resulting policy is evaluated on all learning units observed so far; historical demonstrations are not replayed.

**Model configuration.** All methods except RETAIN use the same pretrained MergeVLA-based architecture, with a shared vision–language backbone and stage-specific action modules, under matched observation, action representation, and evaluation settings. Method-specific adaptation and optimization settings are detailed in Appendix B.

**Baselines.** We compare sequential adaptation methods, continual-merging methods, and an offline reference. **SeqFFT** and **SeqLoRA** provide unconstrained sequential adaptation baselines: SeqFFT updates the shared backbone directly, whereas SeqLoRA parameterizes backbone adaptation with LoRA. **Fixed-Merge** follows the same low-rank adaptation and explicit residual-consolidation protocol as our method but removes both historical-subspace constraints, serving as a direct ablation baseline. **RETAIN** (Yadav et al., 2026) alternates incoming-stage adaptation with parameter merging and carries the resulting merged policy to the subsequent stage. **OPCM** (Tang et al., 2025)

Table 1: Performance on the ten-task LIBERO-Goal stream. Joint is an offline reference excluded from ranking. All values are percentages except BWT, which is reported in percentage points; higher is better.

| Method | Final Avg. | CL-AUC | BWT | New-unit SR | Final Worst |
|---|---|---|---|---|---|
| SeqFFT | 16.5 | 30.4 | -88.3 | 96.0 | 0.0 |
| SeqLoRA | 34.0 | 41.3 | -70.0 | **97.0** | 0.0 |
| Fixed-Merge | 39.0 | 57.9 | -63.3 | 96.0 | 0.0 |
| RETAIN | 0.0 | 14.3 | **-32.2** | 29.0 | 0.0 |
| OPCM | 3.6 | 49.3 | -58.2 | 56.0 | 0.0 |
| **Ours** | **45.5** | **69.2** | -50.0 | 90.5 | 0.0 |
| Joint† | 96.5 | – | – | – | – |

†Joint has simultaneous access to demonstrations from all tasks. A dash indicates an unavailable or inapplicable result.

projects incoming parameter updates with respect to the accumulated update before recursive merging; in our VLA instantiation, OPCM operates on the shared vision–language backbone while stage-specific action modules remain separate. **Joint** trains on demonstrations from all stages simultaneously and is reported only as an offline reference with full historical data access; it is excluded from sequential-method ranking. Detailed baseline adaptations are provided in Appendix C.

**Evaluation metrics.** Let $R_{t,i}$ denote the success rate of learning unit $i$ after stage $t$, where a learning unit is an individual task in the within-suite protocol and a complete suite in the cross-suite protocol. We define the stage-wise average performance over all observed units as $A_t = \frac{1}{t}\sum_{i=1}^{t} R_{t,i}$. We report Final Avg. SR for final-stage average performance, CL-AUC for performance over the complete learning trajectory, BWT for changes in previously acquired skills, New-unit SR for immediate acquisition of incoming units, and Final Worst SR for the weakest final learning unit. Formal definitions are provided in Appendix D. All success-rate metrics are percentages, while BWT is reported in percentage points (pp).

### 5.2 Within-Suite Long-Sequence Continual Merging

We first evaluate **Q1** on the ten-task LIBERO-Goal stream, which applies ten consecutive adaptation–consolidation stages within a single suite.

**Continual performance and retention.** Our method achieves the highest Final Avg. SR of $45.5\%$ and CL-AUC of $69.2\%$ among the evaluated rehearsal-free sequential methods. SeqFFT and SeqLoRA attain high New-unit SRs of $96.0\%$ and $97.0\%$, respectively, but substantially lower Final Avg. SR and strongly negative BWT, indicating that strong immediate acquisition does not prevent substantial forgetting on the Goal-10 stream. Relative to SeqLoRA, our method improves Final Avg. SR and CL-AUC by $11.5$ and $27.9$ pp, respectively. Compared with Fixed-Merge, which follows the same adaptation and consolidation protocol without historical-subspace constraints, our method improves Final Avg. SR, CL-AUC, and BWT by $6.5$, $11.3$, and $13.3$ pp, respectively, while New-unit SR decreases from $96.0\%$ to $90.5\%$. This pattern indicates that the improvement is primarily associated with better retention of previously learned skills rather than stronger immediate acquisition of incoming tasks. Among alternative continual-merging methods, OPCM obtains a Final Avg. SR of $3.6\%$ and a CL-AUC of $49.3\%$ under the evaluated VLA instantiation, substantially below our method on both metrics. RETAIN performs poorly on this long within-suite stream, with a Final Avg. SR of $0.0\%$, a CL-AUC of $14.3\%$, and a New-unit SR of only $29.0\%$. Although its BWT is numerically less negative ($-32.2$ pp), this value should be interpreted together with its weak task acquisition, since poorly acquired tasks leave less performance to be subsequently lost. Overall, these results show that historical-subspace-constrained adaptation improves trajectory-level continual performance and historical retention under repeated consolidation on the Goal-10 stream. Nevertheless, the $0.0\%$ Final Worst SR indicates that long-sequence forgetting is reduced rather than eliminated.

Table 2: Cross-suite continual VLA merging along Spatial → Object → Goal. Suite-specific entries are mean success rates over tasks within each suite. Joint is an offline reference excluded from ranking. All values are percentages; higher is better.

| Method | After Object ↑ | | After Goal ↑ | | | Final Summary ↑ | |
|---|---|---|---|---|---|---|---|
| | Spatial | Object | Spatial | Object | Goal | Final Avg. | Worst Suite Avg. |
| SeqLoRA | 73.8 | **98.0** | 25.8 | 46.8 | **94.4** | 55.7 | 25.8 |
| RETAIN | 80.4 | 79.8 | 57.8 | 18.8 | 26.8 | 34.5 | 18.8 |
| OPCM | 90.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **Ours** | **90.8** | 96.6 | **85.8** | **91.2** | 85.8 | **87.6** | **85.8** |
| Joint† | – | – | 87.4 | 94.6 | 80.2 | 87.4 | 80.2 |

†Joint is evaluated only at the final checkpoint and does not constitute a sequential run.

Table 3: Component ablations on Goal-10. All metrics are percentages except BWT (percentage points); higher is better.

| Variant | Final Avg. | CL-AUC | BWT | New-unit SR | Final Worst |
|---|---|---|---|---|---|
| Fixed-Merge | 39.0 | 57.9 | -63.3 | **96.0** | 0.0 |
| Orth only | 40.5 | 59.8 | -57.2 | 92.0 | 0.0 |
| Subspace-L2 only | 36.0 | 55.3 | -61.1 | 91.0 | 0.0 |
| **Full** | **45.5** | **69.2** | **-50.0** | 90.5 | 0.0 |

### 5.3 Cross-Suite Continual VLA Merging

We next evaluate **Q1** under cross-suite shifts using the Spatial→Object→Goal stream. Table 2 reports suite-average success rates after the Object and Goal stages.

**Overall performance and historical retention.** Our method reaches a Final Avg. SR of $87.6\%$ and a Worst Suite Avg. of $85.8\%$, compared with $55.7\%$ / $25.8\%$ for SeqLoRA and $34.5\%$ / $18.8\%$ for RETAIN. More directly, performance on the Spatial and Object suites after subsequent Goal adaptation decreases by only $5.0$ and $5.4$ pp under our method, whereas SeqLoRA exhibits decreases of $48.0$ and $51.2$ pp and RETAIN shows decreases of $22.6$ and $61.0$ pp. These stage-wise retention patterns are consistent with reduced interference on previously acquired suites. OPCM retains $90.0\%$ on Spatial after the Object stage but obtains $0.0\%$ on Object. After subsequent adaptation to Goal, performance on all three suites drops to $0.0\%$. This pattern indicates that OPCM does not maintain stable cross-suite performance under the evaluated VLA protocol.

**Acquisition trade-off.** Our method achieves $96.6\%$ on the Object suite when it is introduced and $85.8\%$ on the Goal suite at the final stage, compared with $98.0\%$ and $94.4\%$ for SeqLoRA. The improved historical retention is therefore accompanied by moderately reduced performance on incoming suites, illustrating the stability–plasticity trade-off.

### 5.4 Ablation Studies

We conduct Goal-10 component ablations to address **Q2**. All variants share the same adaptation and consolidation protocol and differ only in the historical-subspace constraints: **Fixed-Merge** uses neither constraint, **Orth only** uses Soft Orth, **Subspace-L2 only** uses Subspace-L2, and **Full** combines both.

**Component effects.** Soft Orth alone improves Final Avg. SR, CL-AUC, and BWT over Fixed-Merge. Subspace-L2 alone yields a modest BWT improvement but lower Final Avg. SR, CL-AUC, and New-unit SR than Fixed-Merge, indicating that update-level regularization alone does not provide a favorable stability–plasticity trade-off in this setting. Combining both constraints yields the strongest Final Avg. SR, CL-AUC, and BWT, improving over Fixed-Merge by $6.5$, $11.3$, and $13.3$ pp, respectively. The additional gain obtained when Subspace-L2 is combined with Soft Orth is

Table 4: Deployment storage for the three-stage Spatial→Object→Goal stream. Fixed-Merge and Ours have the same deployment footprint because both consolidate and discard temporary backbone adapters. The "Relative" column is normalized by Separate policies.

| **Deployment strategy** | **Storage (MiB)** | **Relative** |
|---|---|---|
| Separate policies | 9461.45 | $1.00\times$ |
| Continual consolidation (Fixed-Merge / Ours) | 3309.88 | $0.35\times$ |

consistent with complementary gradient-level and update-level historical-subspace control. The improved retention comes at the cost of reduced immediate acquisition: New-unit SR decreases from $96.0\%$ for Fixed-Merge to $90.5\%$ for Full. Moreover, all four variants obtain $0.0\%$ Final Worst SR. Thus, the proposed constraints improve average and trajectory-level continual performance but neither eliminate the stability–plasticity trade-off nor prevent worst-case learning-unit failure.

**Sensitivity to the merge coefficient.** We evaluate nominal export scales $c \in \{0.4, 0.6, 0.8, 1.0\}$, corresponding to effective merge coefficients $\alpha = c^2 \in \{0.16, 0.36, 0.64, 1.0\}$. Among the tested settings, $\alpha = 0.64$ ($c = 0.8$) achieves the highest Final Avg. SR ($87.6\%$) and Worst Suite Avg. ($85.8\%$). Smaller coefficients favor final Spatial performance but yield substantially lower Object and Goal scores, whereas $\alpha = 1.0$ improves the final Goal score at the expense of earlier suites. Complete results are provided in Appendix E.2.

### 5.5 Storage and Deployment Overhead

We address **Q3** by distinguishing deployment storage required at inference time from historical state retained only for future continual adaptation. All reported storage values correspond to the three-stage Spatial→Object→Goal stream.

Continual consolidation requires $3309.88$ MiB at deployment, representing a $65\%$ reduction compared to maintaining separate full policies. Historical-subspace constraints introduce additional state required only for subsequent continual adaptation. Our implementation retains $153.94$ MiB of stage-specific low-rank activation bases for Subspace-L2 and $8027.35$ MiB of dense cumulative projectors for Soft-Orth. Including these training-time components, the total storage footprint is $11491.17$ MiB in the three-stage cross-suite setting. These historical components are not used during policy inference and do not affect deployment storage or inference-time computation. Detailed storage accounting is provided in Appendix F.

## 6 Conclusion

We studied rehearsal-free continual VLA model merging, where consolidated parameters from each stage initialize subsequent adaptation without access to historical demonstrations. We proposed *history-aware subspace-constrained adaptation*, which retains compact historical activation subspaces and uses them to constrain incoming backbone adaptation at two complementary levels. A soft orthogonal gradient constraint attenuates gradient components aligned with a cumulative historical subspace, providing gradient-level control during adaptation. Subspace-L2 regularization penalizes the response induced by the effective backbone update along retained stage-specific historical input subspaces, providing update-level regularization along historical directions. After adaptation, the learned update is consolidated into the shared vision–language backbone, while stage-specific action modules are retained. Experiments on within-suite and cross-suite LIBERO task streams showed that our method achieves the highest final average success rate among the evaluated rehearsal-free approaches, reaching $45.5\%$ on the ten-task LIBERO-Goal stream and $87.6\%$ on the Spatial→Object→Goal stream. Ablation studies further support the benefit of combining the two historical-subspace constraints. Overall, our results show that compact historical activation subspaces can provide effective constraints for rehearsal-free continual VLA model merging. Future work may investigate adaptive constraint weighting based on task similarity, task-compatibility-aware consolidation strategies, and extensions to longer task sequences and broader VLA architectures.

## REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, Laura Smith, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. $\pi_0$: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.010.

Yuxia Fu, Zhizhen Zhang, Yuqi Zhang, Zijian Wang, Zi Huang, and Yadan Luo. Mergevla: Cross-skill model merging toward a generalist vision-language-action agent. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22335–22347, June 2026.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=6t0Kwf8-jrj.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025. URL https://proceedings.mlr.press/v270/kim25c.html.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023. doi: 10.52202/075280-1939. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/8c3c666820ea055a77726d66fc7d447f-Paper-Datasets_and_Benchmarks.pdf.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Daniel Marczak, Bartłomiej Twardowski, Tomasz Trzciński, and Sebastian Cygert. Magmax: Leveraging model merging for seamless continual learning. In Aleš Leonardis, Elisa Ricci, Stefan Roth, Olga Russakovsky, Torsten Sattler, and Gül Varol (eds.), *Computer Vision – ECCV 2024*, pp. 379–395, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-73013-9.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Zihuan Qiu, Lei Wang, Yang Cao, Runtong Zhang, Bing Su, Yi Xu, Fanman Meng, Linfeng Xu, Qingbo Wu, and Hongliang Li. Null-space filtering for data-free continual model merging: Preserving stability, promoting plasticity, 2026. URL https://arxiv.org/abs/2509.21413.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin

Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Sabela Ramos, Sertan Girgin, Léonard Hussenot, Damien Vincent, Hanna Yakubovich, Daniel Toyama, Anita Gergely, Piotr Stanczyk, Raphael Marinier, Jeremiah Harmsen, et al. Rlds: an ecosystem to generate, share and use datasets in reinforcement learning. *arXiv preprint arXiv:2111.02767*, 2021.

Anke Tang, Enneng Yang, Li Shen, Yong Luo, Han Hu, Bo Du, and Dacheng Tao. Merging models on the fly without retraining: A sequential approach to scalable continual model merging. *arXiv preprint arXiv:2501.09522*, 2025.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 23965–23998. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/wortsman22a.html.

Yajat Yadav, Zhiyuan Zhou, Andrew Wagenmaker, Karl Pertsch, and Sergey Levine. Robust fine-tuning of vision-language-action robot policies via parameter merging. In *International Conference on Learning Representations*, volume 2026, pp. 56385–56417, 2026.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11975–11986, October 2023.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, Quan Vuong, Vincent Vanhoucke, Huong Tran, Radu Soricut, Anikait Singh, Jaspiar Singh, Pierre Sermanet, Pannag R. Sanketi, Grecia Salazar, Michael S. Ryoo, Krista Reymann, Kanishka Rao, Karl Pertsch, Igor Mordatch, Henryk Michalewski, Yao Lu, Sergey Levine, Lisa Lee, Tsang-Wei Edward Lee, Isabel Leal, Yuheng Kuang, Dmitry Kalashnikov, Ryan Julian, Nikhil J. Joshi, Alex Irpan, Brian Ichter, Jasmine Hsu, Alexander Herzog, Karol Hausman, Keerthana Gopalakrishnan, Chuyuan Fu, Pete Florence, Chelsea Finn, Kumar Avinava Dubey, Danny Driess, Tianli Ding, Krzysztof Marcin Choromanski, Xi Chen, Yevgen Chebotar, Justice Carbajal, Noah Brown, Anthony Brohan, Montserrat Gonzalez Arenas, and Kehang Han. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023. URL https://proceedings.mlr.press/v229/zitkovich23a.html.

**Algorithm 1** Stage-wise adaptation, export, and historical-state update.

**Require:** Stream $\{(\mathcal{T}_k, \mathcal{D}_k)\}_{k=1}^{K}$; initial policy and historical state; first adapted stage $k_0$; recorded training and export settings
**Ensure:** Consolidated policy, retained action modules, and historical state
**for** $k = k_0, \ldots, K$ **do**
  Attach fresh LoRA factors and initialize the current action module
  Freeze adapted base weights and historical action modules
  Load the stage-specific bases $\{U_j\}_{j<k}$ and cumulative historical basis $Q_{<k}$.
  **while** the stage training loop remains active within its configured limit **do**
    Sample current-stage data from $\mathcal{D}_k$
    Compute the joint objective using the configured loss reductions
    Backpropagate, applying equation 7 through the compatible gradient hooks
    Accumulate gradients and take an optimizer step when due
  **end while**
  Load the selected adapter checkpoint and associated component files
  Export the adapted weights using the checkpoint's merge settings
  Remove LoRA modules from the deployed backbone; retain the action module
  Execute current-unit rollouts and accumulate module-input second moments
  Extract the stage-specific basis $U_k$ from rollout activations
  Update the historical subspace memory
  Do not retain raw activation vectors or use earlier demonstrations for subsequent adaptation
**end for**
**return** The consolidated policy, action modules, and historical state

# A Stage-wise Adaptation and Consolidation

Base weights of LoRA-adapted modules and historical action modules are frozen during adaptation. The trainable state comprises the current LoRA factors, action head, proprioceptive projector, and action queries. The training forward pass includes internal PEFT scaling but not the additional export scale. Selected action-query and action-module files are handled separately from the LoRA weight merge. Algorithm 1 summarizes the recovered operation sequence. Goal-10 starts with an empty historical state. In the cross-suite runs, the Spatial expert supplies the initial policy and adaptation continues with Object and Goal.

# B Experimental Setup and Implementation Details

## B.1 Model Architecture and Adaptation Configuration

We employ the MergeVLA-based modular architecture (Fu et al., 2026), integrating DINOv2 ViT-L/14 and SigLIP ViT-SO/14 Zhai et al. (2023) visual encoders at $224 \times 224$ resolution with a Qwen2.5-0.5B Qwen et al. (2025) language model. The shared backbone processes agent-view and eye-in-hand RGB observations alongside language instructions. An eight-dimensional proprioceptive state vector is encoded via a linear–GELU–linear projector before being fed to an $L_1$-regression action head that predicts eight-step chunks of seven-dimensional end-effector actions.

In our modular framework, each module $\phi_k$ (defined in §3.1) comprises the action head, proprioceptive projector, and associated normalization statistics. The cross-suite protocol maintains one module per suite; Goal-10 maintains one per task. Module selection is determined by the stage identity, which is used internally for routing but not concatenated to the backbone input. For shared-policy baseline configurations, see Appendix C.

The LoRA adapter contains 383 parameter pairs: 381 matrix factor pairs in transformer blocks and two convolutional pairs in visual patch embeddings. LoRA targets include attention and MLP projections in both language and vision backbones, the multimodal projector, and the language model head. Historical gradient constraints (§4.2) apply to the compatible subset.

Table 5: Model and LoRA configuration. All settings are documented by the implementation and validated against saved checkpoints.

| Setting | Value |
|---|---|
| Visual encoders | DINOv2 ViT-L/14; SigLIP ViT-SO/14 |
| Language model | Qwen2.5-0.5B |
| Image resolution | $224 \times 224$ |
| RGB views | Agent-view and eye-in-hand |
| Proprioceptive / action dimensions | 8 / 7 |
| Action chunk length $H$ | 8 |
| LoRA rank $r$ / $\alpha_{\text{LoRA}}$ | 64 / 128 |
| LoRA dropout / bias | 0 / none |
| LoRA initialization | Gaussian $A$; zero $B$ |
| Internal LoRA scaling $s$ | $\alpha_{\text{LoRA}}/r = 2$ |
| Historical-constraint scope | All name- and dimension-compatible modules |

Table 6: Goal-10 task sequence. Instructions follow LIBERO task definitions.

| Stage | Language instruction | Demos |
|---|---|---|
| 1 | open the middle drawer of the cabinet | 50 |
| 2 | put the bowl on the stove | 50 |
| 3 | put the wine bottle on top of the cabinet | 50 |
| 4 | open the top drawer and put the bowl inside | 50 |
| 5 | put the bowl on top of the cabinet | 50 |
| 6 | push the plate to the front of the stove | 50 |
| 7 | put the cream cheese in the bowl | 50 |
| 8 | turn on the stove | 50 |
| 9 | put the bowl on the plate | 50 |
| 10 | put the wine bottle on the rack | 50 |

Goal-10 experiments initialize from the native VLM checkpoint with the configuration above. Cross-suite experiments initialize from the trained Spatial expert and propagate consolidated checkpoints sequentially through Object and Goal stages.

### B.2 Task Streams and Demonstrations

All experiments use LIBERO 0.1.0 (Liu et al., 2023) with official demonstrations in the RLDS format (Ramos et al., 2021), excluding no-operation transitions (no robot motion). Each task provides 50 demonstrations. Continual adaptation uses only current-stage data without rehearsal. Evaluation employs simulator rollouts with fixed initial states.

**Cross-suite protocol.** The task stream follows Spatial→Object→Goal. Each suite contains ten tasks and 500 demonstrations (Table 6). Under the modular protocol, tasks within a suite share one action module and are differentiated by language instructions. The Spatial stage uses a pretrained expert checkpoint.

**Goal-10 protocol.** The ten LIBERO-Goal tasks are introduced sequentially as listed in Table 6. Each stage adds 50 demonstrations. Evaluation after stage $k$ assesses all tasks $1, \ldots, k$.

### B.3 Optimization and Historical-Subspace Hyperparameters

We use AdamW (Loshchilov & Hutter, 2019) with learning rate $2 \times 10^{-4}$, per-device batch size 8, and gradient accumulation steps 2, yielding effective batch size 16 on a single GPU. The policy operates in BF16 precision. Historical second moments $C_k^{(\ell)}$ (Eq. 11) are accumulated in FP64;

Table 7: Training hyperparameters validated from logs and checkpoints. Step counts denote configured maxima.

| Setting | Value |
|---|---|
| Optimizer / learning rate | AdamW / $2 \times 10^{-4}$ |
| Per-device batch / accumulation | 8 / 2 |
| Effective batch size | 16 (single GPU) |
| Policy precision | BF16 |
| Second-moment $C_k^{(\ell)}$ precision | FP64 |
| Historical basis $U_k^{(\ell)}$ precision | FP32 |
| Stage-specific basis rank cap | 16 |
| Cumulative basis construction | Truncated SVD of $[Q_{<k}, U_k]$ |
| Soft-Orth coefficient $\lambda_{\text{orth}}$ | 0.7 (when historical state exists) |
| Subspace-L2 weight $\lambda_{\text{sub}}$ (SOG) | 1000 (Object and Goal) |
| Goal-10 training steps | 10,005 |
| SOG training steps | Object: 20,005; Goal: 40,005 |
| Learning rate schedule | MultiStepLR (milestone: 200,000; $\gamma = 0.1$) |
| AdamW weight decay | 0.01 |
| AdamW $(\beta_1, \beta_2, \epsilon)$ | $(0.9, 0.999, 10^{-8})$ |

stage-specific bases $U_k^{(\ell)}$ are stored in FP32. During regularization, each basis is cast to BF16 for projection computation, then the squared response is reduced to FP32 for loss calculation.

AdamW hyperparameters follow PyTorch 2.2.0 defaults (Paszke et al., 2019). The learning rate schedule milestone exceeds all stage budgets.

**Export scale and effective merge coefficient.** When exporting adapters, both LoRA factors are scaled by coefficient $c$ before merging. For the standard PEFT formulation $\Delta W_{\text{PEFT}} = sBA$ with internal scaling $s$, the export operation yields

$$\Delta W_{\text{export}} = s(cB)(cA) = c^2 \cdot sBA = c^2 \Delta W_{\text{PEFT}}. \tag{10}$$

Thus the effective merge coefficient is $\alpha = c^2$. For the documented setting $c = 0.8$, we obtain $\alpha = 0.64$. Note that $c$ (export scaling), $s$ (internal LoRA scaling $\alpha_{\text{LoRA}}/r$), and $\alpha$ (effective merge strength) are distinct quantities.

### B.4 Historical Activation Collection

Historical statistics are collected via forward hooks during policy rollouts on the current learning unit. Hooks capture module inputs (not outputs) and flatten all leading dimensions into feature vectors arranged as columns of $X_k^{(\ell)} \in \mathbb{R}^{d \times n_k^{(\ell)}}$. The uncentered second moment is accumulated incrementally in FP64:

$$C_k^{(\ell)} = \frac{1}{n_k^{(\ell)}} X_k^{(\ell)} {X_k^{(\ell)}}^\top, \tag{11}$$

where $n_k^{(\ell)}$ is the total number of activation vectors. Both successful and failed rollouts contribute to the statistics.

The cumulative historical basis is updated after each stage according to Eq. (6) and is capped at rank $q_{\text{hist}} = 16$.

Activation collection logs for Goal-10 stages 1–2 each record 20 rollouts. Rollout videos may be saved for diagnostic purposes but are not used in training.

### B.5 Rollout Evaluation Protocol

Rollouts initialize from LIBERO's fixed initial states and execute 10 no-operation steps for simulator stabilization. Policy execution horizons are suite-specific: 220 (Spatial), 280 (Object), and 300

(Goal and Goal-10) steps. Each policy query generates an eight-action chunk executed open-loop. Success is determined by the environment's binary completion signal; reaching the horizon without success constitutes failure.

Each task is evaluated over 20 rollouts for Goal-10 and 50 rollouts for cross-suite experiments. These are repeated trials of a single checkpoint, not independent training runs. Reported success rates are point estimates from these rollout ensembles.

### B.6 Computing Infrastructure and Reproducibility

All experiments run on a single NVIDIA H100 80GB GPU with PyTorch 2.2.0 and PEFT 0.11.1. The LIBERO environment uses seed 0; evaluation rollouts use seed 7 for cross-suite and seed 0 for Goal-10 RETAIN and OPCM methods.

Final reported results correspond to the settings in Table 7.

## C Baseline Adaptations

This section details the baseline configurations used in our comparative evaluation. All modular baselines (SeqFFT, SeqLoRA, Fixed-Merge) share the architecture in §B.1 with method-specific training procedures. RETAIN and OPCM employ distinct implementation strategies described below.

### C.1 Modular Sequential Baselines

**SeqFFT.** Sequential full fine-tuning updates all model parameters at each stage without parameter-efficient adaptations. For Goal-10, we use learning rate $5 \times 10^{-5}$, per-device batch size 8, gradient accumulation steps 2 (effective batch size 16), and 10,005 training steps per stage. Earlier stages use learning rate $2 \times 10^{-4}$ with otherwise identical settings. Both Soft-Orth and Subspace-L2 regularization are disabled. Since no LoRA is used, export scaling does not apply.

**SeqLoRA.** Sequential LoRA training uses rank-64 adapters with $\alpha_{\text{LoRA}} = 128$, following the configuration in Table 5. Training hyperparameters match our method: learning rate $2 \times 10^{-4}$, batch size 8, accumulation steps 2, and 10,005 steps per stage. Historical constraints (Soft-Orth and Subspace-L2) are disabled. Adapters are merged via standard PEFT with effective coefficient $\alpha = 1$ (no scaling).

**Fixed-Merge.** This baseline applies fixed-coefficient scaling at adapter merge time without historical constraints during training. Training configuration matches SeqLoRA: rank-64 LoRA, learning rate $2 \times 10^{-4}$, batch size 8, accumulation steps 2, 10,005 steps, with both regularizers disabled. At each stage $k$, the adapted LoRA factors are scaled by $c = 0.8$ before merging, yielding effective coefficient $\alpha = c^2 = 0.64$ (§B.3).

### C.2 RETAIN: VLA-Level Interpolation

RETAIN (Yadav et al., 2026) performs sequential skill acquisition by interpolating the policy adapted to the current stage with the consolidated policy from the previous stage. We implement RETAIN at the VLA-policy level and use the same formulation in both the Goal-10 and cross-suite experiments. Specifically, after adaptation at stage $k$, the model parameters are consolidated as

$$W_k^{\text{merge}} = 0.5 \cdot W_{k-1}^{\text{merge}} + 0.5 \cdot W_k^{\text{adapt}}. \tag{12}$$

where $W_{k-1}^{\text{merge}}$ denotes the consolidated policy from the previous stage and $W_k^{\text{adapt}}$ denotes the policy adapted to the current-stage data. The interpolation is applied to the VLA policy, including the vision–language backbone and action-related components, using a fixed merge coefficient of 0.5.

### C.3 OPCM: Orthogonal Projection in Weight Space

OPCM (Tang et al., 2025) projects incoming task vectors onto the orthogonal complement of cumulative update directions in weight space. For each stage $k$, we define task vectors relative to a fixed

anchor $W_{\text{anchor}}$ (the stage-0 checkpoint):

$$T_k = W_k^{\text{adapt}} - W_{\text{anchor}}, \qquad M_{k-1} = W_{k-1}^{\text{merge}} - W_{\text{anchor}}. \tag{13}$$

For each eligible weight matrix, OPCM computes the SVD of the cumulative update $M_{k-1} = U\Sigma V^\top$ and projects the incoming update onto these bases. Components corresponding to the top singular values (cumulative energy threshold $0.5$) are zeroed to enforce orthogonality. The projected task vector is combined with the scaled cumulative update and normalized to the mean task-vector magnitude across parameters.

Our OPCM implementation operates on shared VLM parameters, including action query tokens handled within the VLM merge. Separate action heads and proprioceptive projectors are excluded from projection and treated as task-specific modules. The threshold $0.5$ is applied to singular values (not squared values) and is unrelated to merge coefficients in our method. Training uses rank-$64$ LoRA with our historical constraints disabled.

### C.4 Reference Configurations

**Joint training.** Joint training uses simultaneous access to all tasks in the sequence and serves as an offline reference, not a rehearsal-free continual baseline. Training details (data sampling, optimization budget, action-module structure) follow the established MergeVLA protocol (Fu et al., 2026).

**Task-isolated expert ensemble.** This deployment reference assembles independently trained suite experts (Spatial, Object, Goal) with a shared backbone and suite-specific LoRA adapters and action modules. Each adapter uses rank $64$, $\alpha_{\text{LoRA}} = 128$, zero dropout, and the target specification in Table 5. An adapter package includes LoRA weight matrices and action query state; action heads, proprioceptive projectors, and normalization statistics are maintained separately per suite. This configuration represents task-isolated performance without continual learning constraints and is not trained under the sequential protocol.

### C.5 Implementation Consistency and Limitations

All baselines use PyTorch 2.2.0, PEFT 0.11.1, and LIBERO 0.1.0 on NVIDIA H100 GPUs, matching our method's infrastructure (§B.6). Evaluation protocols (rollout budgets, success criteria) are identical across methods.

Key differences from our method:

- **SeqFFT, SeqLoRA, Fixed-Merge:** Disable historical gradient constraints.
- **RETAIN:** Uses shared action modules and accesses historical task performance for checkpoint selection.
- **OPCM:** Operates in weight space rather than gradient space; excludes task-specific heads from projection.
- **Joint, Task-isolated:** Not continual baselines; serve as reference points.

Where multiple implementation variants exist (RETAIN cross-suite), we report the configuration yielding the best performance on development evaluations, following standard practice in continual learning benchmarks.

## D Evaluation Metrics

Let $\mathcal{T}_1, \ldots, \mathcal{T}_K$ be the learning-unit stream, and let $R_{t,i}$ be the success rate of unit $i$ after stage $t$, for $1 \le i \le t \le K$. A unit is a single task in Goal-10 and a complete suite in the cross-suite protocol. The following definitions express success rates in percentage units. For an individual task $u$ evaluated with $N_u$ rollouts,

$$\text{SR}_t(u) = \frac{100}{N_u} \sum_{n=1}^{N_u} \mathbf{1}\{\text{success}_{t,n}(u)\}. \tag{14}$$

In Goal-10, $R_{t,i} = \mathrm{SR}_t(\mathcal{T}_i)$. For a suite-level unit,

$$R_{t,i} = \frac{1}{|\mathcal{T}_i|} \sum_{u \in \mathcal{T}_i} \mathrm{SR}_t(u). \tag{15}$$

The average over all units observed by stage $t$ is

$$A_t = \frac{1}{t} \sum_{i=1}^{t} R_{t,i}. \tag{16}$$

Final performance is summarized by

$$\begin{aligned} \text{Final Avg. SR} &= A_K = \frac{1}{K} \sum_{i=1}^{K} R_{K,i}, \\ \text{Final Worst SR} &= \min_{1 \le i \le K} R_{K,i}. \end{aligned} \tag{17}$$

In the cross-suite protocol, the latter is the worst suite-average score, not the worst individual task across all suites.

Immediate acquisition is summarized by

$$\text{New-unit SR} = \frac{1}{K} \sum_{t=1}^{K} R_{t,t}. \tag{18}$$

For $K > 1$, backward transfer is

$$\mathrm{BWT} = \frac{1}{K-1} \sum_{i=1}^{K-1} (R_{K,i} - R_{i,i}). \tag{19}$$

BWT is measured in percentage points; negative values indicate an average decline from acquisition-time performance.

The discrete continual-learning area under the curve is

$$\text{CL-AUC} = \frac{1}{K} \sum_{t=1}^{K} A_t. \tag{20}$$

This arithmetic mean includes stage 1 and uses no trapezoidal interpolation. CL-AUC and all success-rate metrics are reported in percent. Joint has no sequential success matrix, so trajectory-based metrics are not reported for it.

# E Additional Experimental Results

## E.1 Stage-wise Performance on Goal-10

Figure 2 shows the evolution of the stage-wise average success rate $A_t$ (Appendix D) over the Goal-10 stream. Tables 8–13 provide the corresponding task-level success matrices, with each entry reporting the success rate over 20 evaluation rollouts. Because $A_t$ averages performance over all tasks observed up to stage $t$, the trajectory complements the final-checkpoint metrics by showing how continual performance evolves under repeated adaptation and consolidation.

**Trajectory-level behavior.** The evaluated methods exhibit markedly different degradation patterns as the task sequence grows. SeqFFT and SeqLoRA lose performance on previously seen tasks rapidly during the early stages and remain in a substantially lower performance range thereafter, despite occasional local recoveries. Fixed-Merge preserves considerably more performance than these unconstrained sequential baselines during the first half of the stream, but its trajectory degrades more sharply in later stages. In contrast, our method shows a more gradual decline and maintains

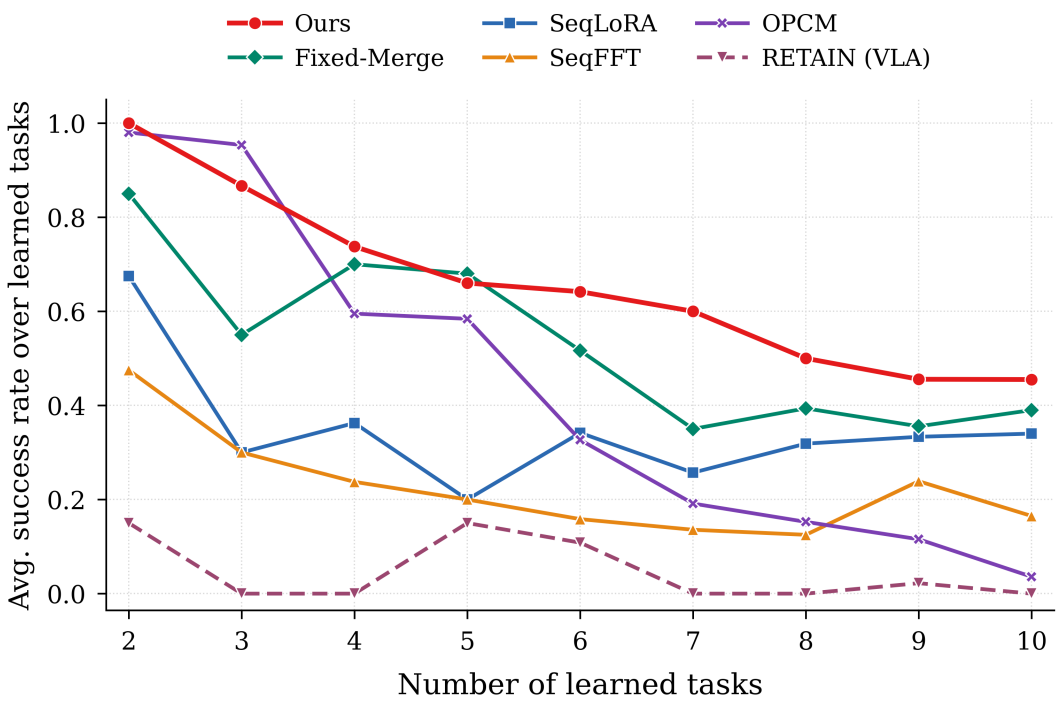


Figure 2: Stage-wise average success rate $A_t$ (Eq. 16) on the Goal-10 stream. The trajectory shows performance on all tasks $1, \ldots, t$ after completing stage $t$. The area under each curve is used to compute CL-AUC (Eq. 20).

the highest or near-highest average success rate over most of the sequence, particularly in the latter half. This pattern is consistent with its higher CL-AUC ($69.2\%$) compared with Fixed-Merge ($57.9\%$), SeqLoRA ($41.3\%$), and SeqFFT ($30.4\%$). The benefit of historical-subspace constraints is therefore not confined to the final checkpoint but persists over a substantial portion of the learning trajectory.

The comparison with Fixed-Merge is particularly informative because both methods follow the same general low-rank adaptation and consolidation paradigm, but Fixed-Merge does not include the proposed historical-subspace constraints. Although Fixed-Merge remains competitive at several early and intermediate stages, its average performance declines more rapidly in the latter half of the stream and reaches $39.0\%$ after stage 10, compared with $45.5\%$ for our method. Together with the corresponding CL-AUC difference, this trajectory shows that scaled consolidation alone provides only partial protection against cumulative degradation, whereas incorporating historical representation information during adaptation improves retention across repeated consolidation stages.

**Behavior of parameter-space continual merging.** OPCM exhibits a distinct early-strong, late-weak trajectory. It achieves $98.0\%$ average success rate after stage 2 and $95.3\%$ after stage 3, but subsequently decreases to $32.7\%$ after stage 6 and $3.6\%$ after stage 10. One plausible interpretation is related to the form of historical information used by OPCM. Its projection operates on an accumulated parameter-space update rather than task-specific historical representation constraints. As more heterogeneous task updates are incorporated, this aggregate reference may increasingly overlap with directions useful for subsequent adaptation, thereby reducing plasticity.

Table 8: Task-level success matrix for SeqFFT on the Goal-10 stream. Each entry reports the success rate (%) over 20 rollouts.

| Stage | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 100 | | | | | | | | | |
| 2 | 0 | 95 | | | | | | | | |
| 3 | 0 | 0 | 90 | | | | | | | |
| 4 | 0 | 0 | 0 | 95 | | | | | | |
| 5 | 0 | 0 | 0 | 0 | 100 | | | | | |
| 6 | 0 | 0 | 0 | 0 | 0 | 95 | | | | |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 95 | | | |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 100 | | |
| 9 | 0 | 0 | 0 | 0 | 0 | 15 | 0 | 100 | 100 | |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 75 | 90 |

Table 9: Task-level success matrix for SeqLoRA on the Goal-10 stream. Each entry reports the success rate (%) over 20 rollouts.

| Stage | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 100 | | | | | | | | | |
| 2 | 35 | 100 | | | | | | | | |
| 3 | 5 | 0 | 85 | | | | | | | |
| 4 | 0 | 0 | 55 | 90 | | | | | | |
| 5 | 0 | 0 | 0 | 0 | 100 | | | | | |
| 6 | 0 | 0 | 5 | 0 | 100 | 100 | | | | |
| 7 | 0 | 0 | 0 | 0 | 30 | 55 | 95 | | | |
| 8 | 0 | 0 | 15 | 0 | 30 | 35 | 75 | 100 | | |
| 9 | 0 | 0 | 0 | 0 | 0 | 55 | 45 | 100 | 100 | |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 45 | 100 | 95 | 100 |

Table 10: Task-level success matrix for Fixed-Merge on the Goal-10 stream. Each entry reports the success rate (%) over 20 rollouts.

| Stage | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 100 | | | | | | | | | |
| 2 | 80 | 90 | | | | | | | | |
| 3 | 40 | 30 | 95 | | | | | | | |
| 4 | 85 | 10 | 95 | 90 | | | | | | |
| 5 | 70 | 45 | 80 | 45 | 100 | | | | | |
| 6 | 60 | 5 | 75 | 0 | 70 | 100 | | | | |
| 7 | 55 | 0 | 35 | 0 | 0 | 65 | 90 | | | |
| 8 | 85 | 0 | 5 | 0 | 0 | 25 | 100 | 100 | | |
| 9 | 20 | 0 | 5 | 0 | 0 | 0 | 100 | 100 | 95 | |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 90 | 100 | 100 | 100 |

Table 11: Task-level success matrix for our method on the Goal-10 stream. Each entry reports the success rate (%) over 20 rollouts.

| Stage | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 100 | | | | | | | | | |
| 2 | 100 | 100 | | | | | | | | |
| 3 | 65 | 95 | 100 | | | | | | | |
| 4 | 5 | 100 | 100 | 90 | | | | | | |
| 5 | 5 | 100 | 100 | 30 | 95 | | | | | |
| 6 | 0 | 95 | 95 | 15 | 90 | 90 | | | | |
| 7 | 0 | 95 | 100 | 0 | 95 | 60 | 70 | | | |
| 8 | 0 | 85 | 60 | 0 | 100 | 50 | 10 | 95 | | |
| 9 | 0 | 80 | 55 | 0 | 90 | 0 | 0 | 95 | 90 | |
| 10 | 0 | 55 | 50 | 0 | 80 | 5 | 0 | 95 | 95 | 75 |

Table 12: Task-level success matrix for OPCM on the Goal-10 stream. Each entry reports the success rate (%) over 20 rollouts.

| Stage | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 100 | | | | | | | | | |
| 2 | 96 | 100 | | | | | | | | |
| 3 | 92 | 100 | 94 | | | | | | | |
| 4 | 18 | 100 | 80 | 40 | | | | | | |
| 5 | 0 | 100 | 66 | 40 | 86 | | | | | |
| 6 | 0 | 92 | 12 | 2 | 82 | 8 | | | | |
| 7 | 0 | 50 | 0 | 0 | 52 | 0 | 32 | | | |
| 8 | 0 | 6 | 0 | 0 | 0 | 0 | 16 | 100 | | |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 4 | 100 | 0 | |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 36 | 0 | 0 |

Table 13: Task-level success matrix for RETAIN on the Goal-10 stream. Each entry reports the success rate (%) over 20 rollouts.

| Stage | T1 | T2 | T3 | T4 | T5 | T6 | T7 | T8 | T9 | T10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 100 | | | | | | | | | |
| 2 | 0 | 30 | | | | | | | | |
| 3 | 0 | 0 | 0 | | | | | | | |
| 4 | 0 | 0 | 0 | 0 | | | | | | |
| 5 | 0 | 0 | 0 | 0 | 75 | | | | | |
| 6 | 0 | 0 | 0 | 0 | 0 | 65 | | | | |
| 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | | |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | | |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 | |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

At the same time, constraining an incoming update relative to an aggregate parameter-space reference does not explicitly preserve the representational constraints of each previously learned task. This combination is consistent with strong performance in the first few stages followed by substantial degradation later in the sequence.

**Behavior of shared-policy interpolation.** RETAIN remains at a substantially lower average success rate throughout most of the evaluated sequence. In the evaluated Goal-10 implementation, RETAIN interpolates not only the shared vision–language parameters but also action-generation components, including the action queries, action head, and proprioceptive projector. Because each Goal-10 stage contains a single manipulation task, current-stage adaptation is driven by a task-specific action distribution. Repeated interpolation of these shared action-generation components may therefore combine parameters that have become specialized to different task-specific behaviors, creating interference across tasks. This provides a plausible explanation for the consistently weak trajectory of RETAIN.

**Retention–acquisition trade-off.** Our method attains a lower New-task SR than SeqLoRA ($90.5\%$ versus $97.0\%$) and Fixed-Merge ($90.5\%$ versus $96.0\%$), but achieves substantially less negative BWT ($-50.0$ versus $-70.0$ and $-63.3$). Together with the trajectory in Figure 2, these results indicate that the improvement in continual performance is primarily associated with better retention of previously acquired tasks rather than stronger immediate acquisition. Nevertheless, $A_t$ continues to decline as more tasks are incorporated, and the final BWT remains negative with a $0.0\%$ Final Worst SR. The proposed constraints therefore mitigate cumulative forgetting under repeated consolidation but do not eliminate long-sequence degradation.

#### E.1.1 Task-Level Success Matrices

The tables below report the task-level success matrices $R_{t,i}$ for the methods evaluated on the Goal-10 stream. Rows index learning stages $t \in \{1, \ldots, 10\}$, and columns index tasks $i$ following the order in Table 6. Each entry $R_{t,i}$ denotes the success rate (%) on task $i$ after completing stage $t$, evaluated over 20 rollouts. Cells are left blank when task $i$ has not yet been introduced ($i > t$).

### E.2 Ablation: Merge Coefficient

We investigate the effect of the merge coefficient on cross-suite performance. Table 14 reports final success rates for different values of the merge scaling parameter $c$ (§B.3). Recall that the effective merge coefficient is $\alpha = c^2$ (Eq. 10).

**Analysis.** The merge coefficient exhibits a clear trade-off between stability and plasticity:

- **Under-scaling ($c = 0.4$, $\alpha = 0.16$):** Preserves Spatial performance (91.8%) but fails to acquire later suites (Object: 0.6%, Goal: 7.6%), yielding low Final Avg. (33.3%). The small effective coefficient limits plasticity on new tasks.
- **Moderate scaling ($c = 0.6$, $\alpha = 0.36$):** Achieves partial acquisition (Object: 58.0%, Goal: 29.8%) with moderate Spatial retention (91.0%). Final Avg. improves to 59.6% but Goal performance remains low.
- **Balanced scaling ($c = 0.8$, $\alpha = 0.64$):** Yields the best overall trade-off with Final Avg. 87.6% and balanced suite performance (Spatial: 85.8%, Object: 91.2%, Goal: 85.8%). This setting is used for the cross-suite results reported in Table 1.
- **Over-scaling ($c = 1.0$, $\alpha = 1.0$):** Maximizes Goal performance (94.0%) but causes severe forgetting on earlier suites (Spatial: 18.4%), resulting in Final Avg. 59.7%. Merging at full coefficient strength ($\alpha = 1.0$) leads to catastrophic forgetting.

Overall, these results indicate that the merge coefficient is an important component of the stability–plasticity trade-off in continual VLA consolidation. This sensitivity analysis characterizes coefficient selection for the full method, while the contribution of the historical-subspace constraints is isolated by the Fixed-Merge comparison in Table 1.

Table 14: Cross-suite final performance as a function of merge scaling parameter $c$. All values are success rates (%). The effective merge coefficient is $\alpha = c^2$.

| $c$ | $\alpha = c^2$ | Suite Success Rate | | | Final Avg. | Final Worst |
|---|---|---|---|---|---|---|
| | | Spatial | Object | Goal | | |
| 0.4 | 0.16 | **91.8** | 0.6 | 7.6 | 33.3 | 0.6 |
| 0.6 | 0.36 | 91.0 | 58.0 | 29.8 | 59.6 | 29.8 |
| 0.8 | 0.64 | 85.8 | **91.2** | 85.8 | **87.6** | **85.8** |
| 1.0 | 1.00 | 18.4 | 66.8 | **94.0** | 59.7 | 18.4 |

# F Storage and Deployment Analysis

We analyze the storage footprint and deployment trade-offs of our consolidation approach compared to task-isolated baselines. Storage requirements comprise two components:

1. **Deployment storage:** model parameters required for inference (§F.1).
2. **Training state:** historical subspace bases and auxiliary structures needed for continual adaptation but not for inference (§F.2).

## F.1 Deployment Storage

**Storage model.** We compare three deployment strategies for a $K$-stage continual learning system:

1. **Separate policies:** maintain $K$ independent full checkpoints.
2. **Task-isolated LoRA:** share one backbone with $K$ LoRA adapters and $K$ task-specific action modules.
3. **Consolidated (Ours):** merge adapters into a single backbone, retaining only $K$ task-specific action modules.

Let $s_{\text{base}}$ denote the shared backbone size, $s_{\text{head}}$ the per-task action module size (action head + proprioceptive projector + action queries), and $s_{\text{ada}}$ the LoRA adapter size (low-rank weight matrices). For separate full policies, $s_{\text{full}} = s_{\text{base}} + s_{\text{head}}$ denotes a complete checkpoint. The deployment storage for $K$ tasks is:

$$\begin{aligned} S_{\text{sep}}(K) &= K \cdot s_{\text{full}} = K(s_{\text{base}} + s_{\text{head}}), \\ S_{\text{iso}}(K) &= s_{\text{base}} + K(s_{\text{head}} + s_{\text{ada}}), \\ S_{\text{cons}}(K) &= s_{\text{base}} + K \cdot s_{\text{head}}. \end{aligned} \tag{21}$$

Both our method and Fixed-Merge use the same consolidated deployment structure. The storage saving relative to task-isolated LoRA comes from adapter merging, not from the historical constraints.

**Measured sizes.** For the three-suite cross-suite protocol (Spatial, Object, Goal), we measure:

- Consolidated backbone: $s_{\text{base}} = 2{,}389.18$ MiB
- Per-suite action module: $s_{\text{head}} = 306.90$ MiB
- Total LoRA adapters (3 suites): $594.58$ MiB (average $198.19$ MiB per suite)

Normalization statistics ($\approx 0.01$ MiB) are negligible and omitted from the analysis.

Table 15 shows the total deployment storage for $K = 3$ stages.

**Analysis.** Consolidation reduces deployment storage by $594.58$ MiB (15.2%) compared to task-isolated LoRA, and by $6{,}151.57$ MiB (65.0%) compared to separate full policies. The saving scales linearly with the number of stages $K$:

$$S_{\text{iso}}(K) - S_{\text{cons}}(K) = K \cdot s_{\text{ada}}, \tag{22}$$

Table 15: Deployment storage for three-suite cross-suite continual learning. All measurements include parameters required for inference. Relative storage is normalized to the separate policies baseline.

| Deployment Strategy | Storage (MiB) | Relative |
|---|---|---|
| Separate policies ($3 \times s_{\text{full}}$) | 9,461.45 | $1.00\times$ |
| Task-isolated LoRA | 3,904.46 | $0.41\times$ |
| Consolidated (Ours) | 3,309.88 | $0.35\times$ |
| *Breakdown for Consolidated:* | | |
| Shared backbone | 2,389.18 | — |
| Action modules ($3 \times 306.90$) | 920.70 | — |

where $s_{\text{ada}} \approx 198.19$ MiB is the average adapter size per suite. For long task sequences (e.g., $K = 10$ in Goal-10), consolidation saves approximately $1{,}982$ MiB compared to task-isolated LoRA deployment, reducing memory requirements for resource-constrained robotic systems.

### F.2 Historical Training State

Continual adaptation retains stage-specific bases $\{U_j^{(\ell)}\}$ for Subspace-L2 regularization and a cumulative historical basis $Q_{<k}^{(\ell)}$ for Soft Orth. This state is needed only for training stage $k+1$ and beyond, not for policy inference.

**Orthonormal basis representation.** At stage $k$, each adapted layer $\ell$ has a stored orthonormal basis $U_k^{(\ell)} \in \mathbb{R}^{d_{\text{in}}^{(\ell)} \times q_k^{(\ell)}}$, where $q_k^{(\ell)} \leq 16$ is the rank determined by the cumulative energy threshold (§B.3).

Let $\mathcal{L}_k$ denote the set of adapted layers at stage $k$. The total basis storage after $K$ stages is:

$$S_{\text{basis}}(K) = \sum_{k=1}^{K} \sum_{\ell \in \mathcal{L}_k} 4 \cdot d_{\text{in}}^{(\ell)} \cdot q_k^{(\ell)} \text{ bytes}, \tag{23}$$

where $4$ bytes accounts for FP32 storage.

For the three-suite experiment, the serialized basis file contains $1{,}788$ FP32 tensors ($596$ layers $\times$ $3$ stages) with tensor data $153.32$ MiB and metadata overhead $0.62$ MiB, totaling $153.94$ MiB.

**Dense projector caching.** To accelerate training, the implementation pre-computes dense orthogonal projectors $P_k^{(\ell)} = I - U_k^{(\ell)}(U_k^{(\ell)})^\top \in \mathbb{R}^{d_{\text{in}}^{(\ell)} \times d_{\text{in}}^{(\ell)}}$ and stores them on disk. For three suites with $596$ layers, the dense projector file occupies $8{,}027.35$ MiB.

Table 16 summarizes all storage components.

### F.3 Storage–Performance Trade-off

We compare deployment performance between task-isolated LoRA (a strong baseline without consolidation) and our consolidation approach.

**Experimental setup.** Both configurations are evaluated on cross-suite with $50$ rollouts per task:

- **Task-isolated:** independently trained suite experts with separate LoRA adapters and action modules (§C).
- **Consolidated (Ours):** final merged backbone with suite-specific action modules.

Table 17 reports the suite-level success rates.

**Analysis.** Consolidation achieves $15.2\%$ deployment storage reduction at the cost of $9.6$ percentage points (pp) Final Avg. and $9.2$pp Final Worst performance degradation.

Table 16: Storage breakdown for cross-suite continual learning with three stages. Deployment components are required for inference. Training state is needed only for adapting to subsequent stages.

| Component | Storage (MiB) | Purpose |
|---|---|---|
| *Deployment (inference):* | | |
| Consolidated backbone | 2,389.18 | Inference |
| Action modules ($3\times$) | 920.70 | Inference |
| *Training state:* | | |
| Orthonormal bases $\{U_k^{(\ell)}\}$ | 153.94 | Continual training |
| Dense projector cache | 8,027.35 | Training acceleration |
| Total deployment | 3,309.88 | — |
| Total with training state | 11,491.17 | — |

Table 17: Storage–performance trade-off for cross-suite deployment. Each suite score averages ten tasks with 50 rollouts per task. Task-isolated represents a strong baseline without consolidation.

| Deployment | Storage (MiB) | Suite Success Rate (%) | | | Final Avg. | Final Worst |
|---|---|---|---|---|---|---|
| | | Spatial | Object | Goal | | |
| Task-isolated | 3,904.46 | 98.0 | 98.6 | 95.0 | 97.2 | 95.0 |
| Consolidated (Ours) | 3,309.88 | 85.8 | 91.2 | 85.8 | 87.6 | 85.8 |
| Difference | $-594.58$ ($-15.2\%$) | $-12.2$ | $-7.4$ | $-9.2$ | $-9.6$ | $-9.2$ |

This trade-off arises from task interference during merging:

- **Task-isolated:** maintains independent parameter spaces for each suite, avoiding cross-task interference but requiring more storage.
- **Consolidated:** merges all adaptations into shared parameters, reducing storage but introducing representational conflicts that our historical constraints mitigate but do not fully eliminate.

For applications prioritizing storage efficiency (e.g., on-robot deployment with limited memory), our consolidation approach offers a favorable trade-off. For applications prioritizing maximum performance (e.g., cloud-based services with ample storage), task-isolated deployment may be preferred despite higher storage costs.

## AI Use Disclosure

Generative AI tools were used to assist with literature discovery, research brainstorming and discussion of experimental methodology, as well as drafting and improving the readability of parts of the manuscript. All AI-assisted content, including technical descriptions, claims, citations, and experimental discussions, was reviewed and verified by the authors. The authors take full responsibility for the final content of this work.

## Reproducibility Statement

The main text and appendices specify the continual-learning protocol, model architecture, optimization settings, baseline adaptations, evaluation metrics, and storage accounting. For the reported implementation, Soft-Orth uses cumulative historical subspaces represented by dense orthogonal-complement projectors, whereas Subspace-L2 uses stage-specific rank-capped activation bases. We report the corresponding training, evaluation, merge/export, and historical-state configurations in the appendix. No historical demonstrations are retained or replayed during subsequent adaptation.