R^2-WAM: FUNCTIONAL REUSE AND REFINEMENT OF HISTORICAL ADAPTATIONS FOR WAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

World-Action Models (WAMs) deployed over long horizons must be repeatedly adapted as tasks and scenes change, yet each adaptation remains tied to its source task. We ask whether saved adapters can serve as reusable experience for new few-shot tasks. Because WAMs route world information, task context, and action generation through separate pathways, a target may benefit from different historical adapters along each pathway, yet still need capabilities absent from the bank. We propose R^2-WAM, built on a function-structured adapter with video, context, and action routes. It ranks historical residuals route by route with first-order target-loss scores, learns task-specific compositions, and adds a group-shared new residual with task-specific exposure for missing capabilities. It needs only saved adapters, not historical data. On 35 held-out LIBERO tasks with three demonstrations each, R^2-WAM reaches 72.2% success, exceeding a fresh adapter by 14.3 percentage points (pp), pooled-data initialization by 6.1 pp, and random historical support by 6.5 pp. Alone, the structured adapter outperforms LoRA by 5.2 pp on LIBERO-Object and 14.0 pp on RoboCasa365 with 80 demonstrations.

1 INTRODUCTION

Robot policies have evolved from vision-language-action (VLA) models (Brohan et al., 2023; Kim et al., 2024; Black et al., 2026) to World-Action Models (WAMs) that jointly model environment dynamics and actions (Ye et al., 2026b; Yuan et al., 2026). As embodied pretraining approaches a million robot trajectories or tens of thousands of GPU-hours (Kim et al., 2024; NVIDIA et al., 2025), stronger priors enable rapid few-shot adaptation and, increasingly, new-task execution from in-context demonstrations without parameter updates (Sridhar et al., 2025; Chen et al., 2025; Zhou et al., 2026), a trend also emphasized in recent industry reports (Skild AI, 2026; Generalist Team, 2026). Deployment on a specific robot, however, still requires parameter updates: new embodiments, scenes, and operational requirements keep arising, and capabilities acquired in context are not retained once the context is gone. Each such update produces an adapter that is typically kept only for its source task. As deployment continues, these adapters accumulate. This raises a natural question: *can saved adaptations serve as reusable experience when the robot faces a new task with only a few demonstrations?*

We study this question in a long-term deployment setting. Such settings are common in practice: a robot on a production line must handle new part variants, and a warehouse robot faces a changing set of items, while the embodiment itself stays largely fixed. A robot is first onboarded with easily collected basic-operation demonstrations, which train a shared anchor policy. As related tasks arrive, they are organized into groups that share an operation template, such as placing different objects into the same kind of receptacle. One adapter is trained per group from the same anchor, and the adapters form a historical bank. When a new group of target tasks arrives with a few demonstrations each, the question becomes how to adapt from the anchor while drawing on the bank.

The most direct approach reuses whole historical adapters, as LoRAHub does for language tasks by composing a frozen LoRA bank from a few examples (Huang et al., 2024). Whole-adapter reuse, however, assumes that a single historical source serves all parts of the target task. WAMs make this assumption unnecessary. Unlike typical VLAs, where visual and language conditioning are fused into one policy representation, WAMs route world information, task context, and action generation through explicit pathways. A target can therefore draw on different historical adapters

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.

along each pathway. On Long-B, for example, the video route draws most on an adapter trained on spatial book-to-shelf placement, whereas the context route draws on one trained on mug-to-plate spatial relations (Figure 8). Under the same candidate budget, composing adapters route by route outperforms whole-adapter composition by 10.2 pp (Section 5.4.2).

Reuse alone is not sufficient either. Language tasks can often be covered by recombining existing modules, but embodied targets introduce new object interactions, spatial relations, and operation sequences that no saved adapter may provide. Continual-learning methods such as CLARE (Römer et al., 2026) and DMPEL (Lei et al., 2026) expand their modules as tasks arrive sequentially, but they make each reuse or expansion decision within a system shaped by earlier tasks. We instead consider a target group that adapts independently from the shared anchor and a frozen bank, and that must both reuse what the bank offers and learn what it lacks.

We propose ***R²-WAM***, which combines functional **R**euse and target-group **R**efinement. It builds on a function-structured adapter that places trainable modules at the video, context, and action interfaces of a frozen WAM, so that all historical adapters share aligned routes. Functional reuse scores each historical residual route by route by how much it reduces the target action loss to first order, keeps the top candidates, and learns a task-specific composition over them. Target-group refinement then adds one new residual shared by the group and scales it by a learned task-specific *exposure*, so that new capacity does not disrupt useful historical support. Our main contributions are as follows:

- We formulate history-aware cross-task few-shot adaptation for WAM deployment and build a LIBERO protocol with onboarding, a 15-adapter historical bank, and held-out target groups.
- We place a function-structured adapter at a frozen WAM's video, context, and action interfaces; with 80 demonstrations, it beats LoRA by 5.2/14.0 pp on LIBERO-Object/RoboCasa365.
- We propose R^2-WAM, which composes historical residuals route-wise and adds a group-shared residual with task-specific exposure for capabilities absent from the bank. It uses only saved adapters and target demonstrations, allowing historical data to be discarded after adapter training.
- On 35 held-out LIBERO tasks with three demonstrations each, R^2-WAM reaches 72.2% success: +14.3 pp over a fresh adapter, +6.1 pp over pooled initialization, and +6.5 pp over random support.

2 Background

2.1 World-Action Models

A World-Action Model jointly predicts a future visual representation and continuous action chunk from observation, instruction, and robot state, typically through interacting video and action pathways. WAMs differ in how future visual information affects action prediction. They may jointly denoise future video and actions with direct cross-modal interaction (Ye et al., 2026b), condition inverse dynamics on a predicted future representation (Li et al., 2026), or causally block future-to-action flow so actions use only current information. Under causal blocking, future dynamics can be action-conditioned (Ye et al., 2026a) or modeled in a world pathway parallel to action prediction (Yuan et al., 2026). Since actions do not read future visual tokens in these architectures, future-video generation can be omitted at deployment.

These causal structures also determine how world and action supervision are coupled. WAMs commonly learn generated-variable velocity fields with flow matching, where $\mathcal{L}_{\mathrm{FM}} = \mathbb{E}[\|\hat{\mathbf{v}} - \mathbf{v}\|_2^2]$. Joint video-action models directly supervise the combined velocity $\mathbf{v}_{va} = [\mathbf{v}_v; \mathbf{v}_a]$, while separated pathways typically optimize

$$\mathcal{L}_{\mathrm{WAM}} = \lambda_{\mathrm{video}}\mathcal{L}_{\mathrm{FM-video}} + \lambda_{\mathrm{action}}\mathcal{L}_{\mathrm{FM-action}}. \tag{1}$$

When future-to-action flow is blocked, action learning is decoupled from future visual targets, leaving video supervision as an independent world-modeling objective. This structure matches our setting: we use the full video-action objective during robot onboarding to establish the base world-action representation, and optimize only the action objective for downstream adaptation.

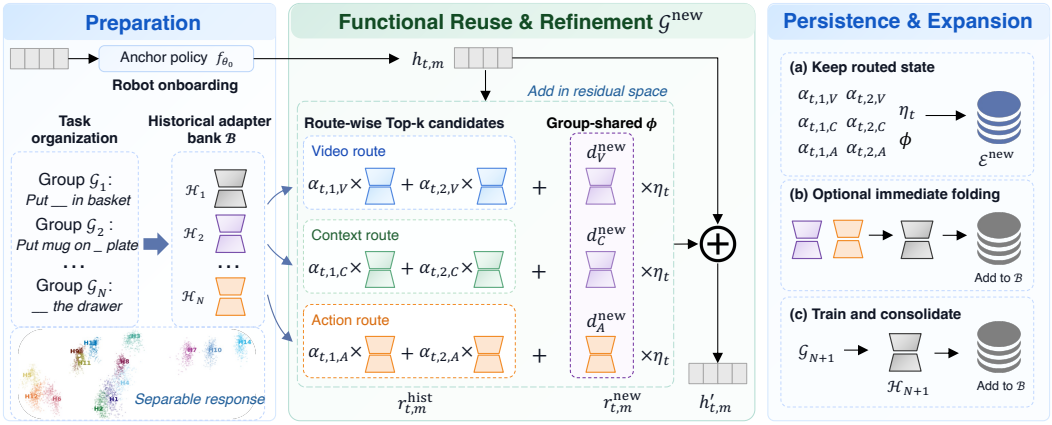


Figure 1: **R^2-WAM deployment lifecycle.** Task-group adapters build the historical bank from the anchor. For few-shot targets, R^2-WAM recombines bank residuals across video, context, and action routes, learns missing capabilities, and supports folding or group-level consolidation.

2.2 Historical Modular Reuse and Expansion

Parameter-efficient fine-tuning (PEFT) keeps a shared backbone f_{θ_0} fixed and learns task- or group-specific parameters H_i. Accumulated modules form an adapter bank $\mathcal{B} = \{H_i\}_{i=1}^{N}$, allowing learned parameters to be reused beyond their source tasks.

Deployment-time routing. Historical modules can be selected after training according to the current task or input. Existing approaches switch experts from task identity or language instructions (Luo et al., 2026), retrieve adapters by semantic similarity (Kuzmenko & Shvai, 2026), or, as in CLARE, select the adapter with the lowest discriminator reconstruction error on current hidden features at each expandable layer (Römer et al., 2026).

Adaptation-time reuse. Historical modules can also directly support learning a new target. LoRAHub optimizes a static composition of frozen LoRA modules from a few target examples (Huang et al., 2024), closely matching the cross-task adaptation setting considered here. This differs from continual learning, where tasks arrive sequentially and each reuse or expansion decision is made within a modular system already shaped by previous tasks (Lei et al., 2026).

Reuse granularity and capacity expansion. Historical experience may be reused at different granularities, from whole adapters to mixtures across layers or functional submodules. An insufficient bank may be expanded. DMPEL adds one expert per task and updates a shared router, replaying historical context-to-coefficient pairs to preserve earlier routing. CLARE estimates novelty from historical reconstruction statistics to choose reuse or expansion.

3 Problem Setting and Formulation

We study history-aware cross-task few-shot adaptation with a shared anchor and a frozen historical adapter bank. With both fixed, target groups reuse bank residuals and learn missing capabilities from demonstrations. Section 6 covers persistence, folding, and bank expansion.

Robot Onboarding and Task Organization. When a new model architecture, robot configuration, or deployment scenario prevents direct reuse of an embodied pretrained policy, basic-operation demonstrations train a robot-specific anchor f_{θ_0}. Its video branch inherits a pretrained video generator, and parameter interpolation initializes the action branch. Eq. (1) establishes basic action generation and world-action correspondence without needing to cover future tasks. Many embodied manipulation tasks can be viewed as instantiations of a shared operation template, with task-specific entities, spatial arguments, and scenes filling its slots. We therefore organize tasks sharing an operation template into groups $\mathcal{G}_i = \{\tau_j\}_{j=1}^{M_i}$, with individual tasks differing in their slot instantiations. Stable groups serve as units for accumulating and retaining long-term parameterized experience.

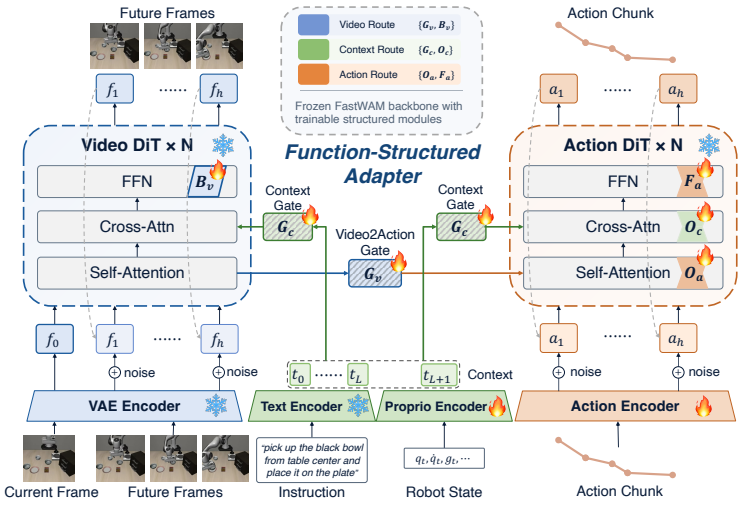


Figure 2: **Function-structured adapter in FastWAM.** Six trainable modules on the frozen backbone form video, context, and action routes for world information, task conditioning, and action transformation. Identity or zero-effect initialization preserves the base function and causal mask.

Function-Structured Adaptation Space. Each historical group $\mathcal{G}_i$ learns a single adapter H_i from the shared anchor in a common function-structured space, capturing adaptation shared across its template instantiations. We define $\mathcal{R} = \{V, C, A\}$ for world information, task context, and action transformation, and write $H_i = \{d_{i,m}^{\text{hist}}\}_{m\in\mathcal{R}}$. Shared route definitions and interfaces align same-route residuals across adapters for comparison and composition. Historical adapters are trained only with $\mathcal{L}_{\text{FM-action}}$ and then frozen into $\mathcal{B} = \{H_i\}_{i=1}^{N}$. Appendix D shows stable, separable adapter-specific response signatures, supporting differentiated parameterized experience in this shared space.

History-Aware Target Group Adaptation. A target group $\mathcal{G}^{\text{new}} = \{\tau_t\}_{t=1}^{M}$ contains tasks sharing an operation template, each with few demonstrations $\mathcal{D}_t$ and no assumed alignment to a single historical group. Different functions may draw from distinct historical sources, while required capabilities may be absent from the bank. We therefore decompose adaptation for task t and route m as

$$\Delta_{t,m} = r_{t,m}^{\text{hist}} + r_{t,m}^{\text{new}}. \tag{2}$$

Functional reuse composes $r_{t,m}^{\text{hist}}$ from bank residuals, while target-group refinement learns $r_{t,m}^{\text{new}}$ for missing capabilities. With f_{θ_0} and $\mathcal{B}$ frozen, we optimize the historical composition and new correction on target demonstrations under action supervision, yielding a bank-dependent routed state.

4 Method

Given a target group $\mathcal{G}^{\text{new}} = \{\tau_t\}_{t=1}^{M}$ with demonstrations $\{\mathcal{D}_t\}_{t=1}^{M}$, frozen anchor f_{θ_0}, and bank $\mathcal{B} = \{H_i\}_{i=1}^{N}$, R^2-WAM parameterizes the two corrections in Eq. (2). For task τ_t and route $m \in \mathcal{R}$,

$$h'_{t,m} = h_{t,m} + \sum_{i\in S_m} \alpha_{t,i,m} d_{i,m}^{\text{hist}}(h_{t,m}) + \eta_t\, d_m^{\text{new}}(h_{t,m}; \phi_m). \tag{3}$$

Here, $h_{t,m}$ is the representation at the current adaptation interface after any upstream adapted computation. Stage 1 selects the route-wise candidate set S_m and learns task-specific historical composition $\alpha_{t,i,m}$. Stage 2 learns the group-shared new residual parameters ϕ_m and task-specific exposure η_t, while jointly readjusting $\alpha_{t,i,m}$. The anchor and historical bank remain frozen throughout.

4.1 Function-Structured Adapters

FastWAM's causal dual-stream backbone blocks future-video-to-action flow, enabling action-only adaptation (Section 2.1). Six trainable modules form the video, context, and action routes $\{G_v, B_v\}$, $\{G_c, O_c\}$, and $\{F_a, O_a\}$ for world information, task context, and action transformation (Figure 2). Identity or zero-effect initialization preserves the base function and aligned adapter interfaces.

Routes follow functional roles. G_v belongs to the video route because it modulates visual information read by action queries, although it lies on the action-query-to-video-value path.

Identity-centered value gating. G_v and G_c apply identity-centered multiplicative gates.

$$\begin{aligned} G_v \quad & g_v = 2\sigma(\ell_v(X_v;\phi_{G_v})), \quad \widetilde{V}_{v\to a} = g_v \odot V_v, \\ G_c \quad & g_c^b = 2\sigma\big(\ell_c^b(V_{\mathrm{ctx}}^b;\phi_{G_c})\big), \quad \widetilde{V}_{\mathrm{ctx}}^b = g_c^b \odot V_{\mathrm{ctx}}^b, \quad b \in \{v,a\}. \end{aligned} \tag{4}$$

Here, ℓ_v and ℓ_c^b are gate-logit networks, and zero-initialized final projections set $g_v = g_c^b = 1$, making the identity-relative residual $(g-1)\odot V$ in Eq. (3) initially zero. G_v gates video values for action queries while preserving the causal mask, whereas G_c gates context values in both branches.

Additive feature calibration. B_v adds a correction after the frozen video FFN.

$$\widetilde{F}_v = F_v + W_\beta \,\mathrm{LN}(H_v). \tag{5}$$

A zero-initialized W_β additively calibrates task-dependent world features with no initial effect.

Zero-effect residual adaptation. We apply a bottleneck residual after action-branch aggregation.

$$\mathcal{A}(H) = W_{\mathrm{up}}\,\mathrm{SiLU}(W_{\mathrm{down}}\,\mathrm{LN}(H)), \qquad \widetilde{H} = H + \mathcal{A}(H). \tag{6}$$

Zero-initialized W_{up} makes the residual initially inactive. We instantiate this form as O_c, F_a, and O_a for context readout, the action FFN, and action self-attention output. These modules are not independent experts, but jointly define the route-level residual in Eq. (3) with other modules on the same route. Historical and new residuals share this parameterization.

4.2 Stage 1: Target-Loss-Driven Functional Reuse

Functional reuse turns frozen bank residuals into target-aligned feature corrections. Target demonstrations first rank them route-wise, then learn task-specific compositions over selected candidates without a semantic retriever or cross-task neural router.

4.2.1 Route-Wise Historical Transferability

We normalize task-dependent action-loss scales by anchor loss before group aggregation.

$$\mathcal{L}_{\mathrm{FM-action}}^{\mathrm{rel}}\left(\{\alpha_t\}_{t=1}^M\right) = \frac{1}{M}\sum_{t=1}^{M} \frac{\mathcal{L}_{\mathrm{FM-action}}(\mathcal{D}_t;\alpha_t)}{\mathrm{stopgrad}\left[\mathcal{L}_{\mathrm{FM-action}}(\mathcal{D}_t;0)\right]+\epsilon}. \tag{7}$$

Here, $\alpha_t = 0$ disables all historical residuals and defines the anchor-relative reference, while the denominator normalizes task scales for both selection and composition. For group-shared probing, we temporarily tie $\alpha_t = \tilde{\alpha}$ across tasks. These probes are distinct from the task-specific composition variables optimized later. The first-order transferability score for adapter i on route m is

$$q_{i,m} = -\left.\frac{\partial}{\partial \tilde{\alpha}_{i,m}} \mathcal{L}_{\mathrm{FM-action}}^{\mathrm{rel}}\left(\{\alpha_t = \tilde{\alpha}\}_{t=1}^M\right)\right|_{\tilde{\alpha}=0}. \tag{8}$$

A small increase $\delta > 0$ in only $\tilde{\alpha}_{i,m}$ changes the objective by $-\delta q_{i,m} + O(\delta^2)$, so larger positive $q_{i,m}$ indicates stronger first-order improvement from the shared anchor reference. Route-wise ranking gives the group-shared candidate set $S_m = \mathrm{TopK}_i(q_{i,m}, k)$ for composition. The task-specific variant uses untied probes, giving $q_{t,i,m}$ and $S_{t,m} = \mathrm{TopK}_i(q_{t,i,m}, k)$ (Section 5.4.1).

4.2.2 Historical Residual Composition

The first-order score only proposes candidates. With S_m fixed, target demonstrations optimize task-specific compositions under $\mathcal{L}_{\mathrm{FM-action}}^{\mathrm{rel}}$. We factor each coefficient into route strength and candidate allocation,

$$\gamma_{t,m} = 2\sigma(\rho_{t,m}), \qquad w_{t,i,m} = \frac{\exp(z_{t,i,m})}{\sum_{j\in S_m}\exp(z_{t,j,m})}, \qquad \alpha_{t,i,m} = \gamma_{t,m} w_{t,i,m}. \tag{9}$$

Here, $\gamma_{t,m}$ sets the total historical strength on route m, while $w_{t,i,m}$ distributes it across candidates. Thus, $\alpha_{t,i,m} \geq 0$, $\sum_{i\in S_m}\alpha_{t,i,m} = \gamma_{t,m} \in (0,2)$, with support restricted to S_m.

During composition, the anchor and historical adapters remain frozen, candidate sets stay fixed, and the new residual is disabled. We optimize $\rho_{t,m}$ and $z_{t,m} = \{z_{t,i,m}\}_{i \in S_m}$ by solving

$$\{\rho_{t,m}^{(0)}, z_{t,m}^{(0)}\} = \underset{\{\rho_{t,m}, z_{t,m}\}}{\arg\min} \mathcal{L}_{\mathrm{FM-action}}^{\mathrm{rel}} \left(\{\alpha_t(\rho_t, z_t)\}_{t=1}^{M} \right). \tag{10}$$

Starting from $\gamma_{t,m} = 1$ and $w_{t,i,m} = 1/k$, optimization yields $\alpha_t^{(0)}$, which instantiates the historical correction in Eq. (3) and initializes Stage 2.

4.3 Stage 2: Target-group refinement

Functional reuse covers only capabilities represented in the bank, so each target group adds one group-shared residual adapter $\phi = \{\phi_m\}_{m \in \mathcal{R}}$. Zero-effect initialization from Section 4.1 preserves the Stage 1 composition. Task-specific exposure $\eta_t = \eta_{\mathrm{init}} + (1 - \eta_{\mathrm{init}})\sigma(s_t)$ controls this capacity through a trainable logit s_t shared across all routes of task t, with $\eta_{\mathrm{init}} = 0.1$. Its low initial value limits disruption, while optimization adapts residual strength to each task.

Stage 2 replaces the normalized Stage 1 objective with the original action loss and equal task weights, jointly optimizing historical composition, the group-shared new residual, and task-specific exposure,

$$\left(\{\rho_{t,m}^*, z_{t,m}^*\}, \phi^*, \{s_t^*\}\right) = \underset{\{\rho_{t,m}, z_{t,m}\}, \phi, \{s_t\}}{\arg\min} \frac{1}{M} \sum_{t=1}^{M} \mathcal{L}_{\mathrm{FM-action}} \left(\mathcal{D}_t; \alpha_t, \phi, \eta_t \right). \tag{11}$$

Because Stage 1 learns the historical composition without the new residual, Stage 2 keeps it trainable for joint readjustment. Together, the optimized corrections realize Eq. (2).

5 Experiments

We evaluate data-budget adaptation in Section 5.2, historical reuse and learned compositions in Section 5.3, and key design choices in Section 5.4.

5.1 Experimental Setup

Overall protocol. All experiments use FastWAM (Yuan et al., 2026), matched target data, and closed-loop success. Fresh baselines match Stage 2, while R^2-WAM also selects and composes historical support. Appendix A details the protocol.

Data-budget evaluation. A LIBERO Spatial anchor adapts to Object at six budgets spanning 20 to all 454 demonstrations, while a RoboCasa365 anchor pretrained on composite and atomic data adapts to held-out ArrangeBreadBasket with 20 or 80 demonstrations.

History-aware evaluation. Our long-term LIBERO protocol starts from 500 LIBERO-90 onboarding demonstrations spanning 10 manipulation primitives and interaction dynamics. Fifteen groups based on reusable object-target relations, spatial relations, and operation structures yield one bank adapter each. We reorganize 35 held-out Spatial, Object, Goal, and Long tasks into six operation-structured groups with 3 demonstrations and 50 rollouts per task.

5.2 Adaptation across Data Budgets

Under the Section 5.1 data-budget protocol, we compare the function-structured adapter with full fine-tuning, tuned LoRA, and a parameter-controlled variant.

The structured adapter gains most under limited data. On LIBERO Spatial-to-Object in Figure 3(a), it reaches 76.6% with 20 demonstrations and 97.2% with 80, versus 71.4%/80.6% for full fine-tuning and 71.8%/92.0% for LoRA. With more data, full fine-tuning catches up and all methods approach saturation. The structure therefore reaches high success sooner under limited data without raising the data-rich ceiling.

Parameter count does not explain the scaling gap. LoRA plateaus around 91.0–94.0% after 80 demonstrations, while the structured adapter reaches 97.2% at 80. Compressed from about 342M to

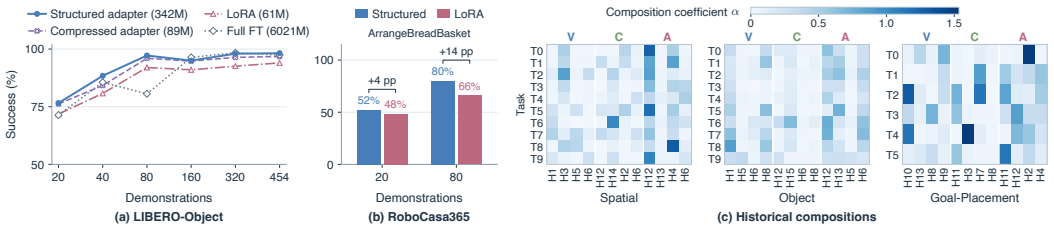


Figure 3: **Data-budget adaptation and historical compositions.** (a) LIBERO-Object methods share a Spatial anchor. (b) RoboCasa365 compares the structured adapter with LoRA. (c) Task-specific Top-4 coefficients for Spatial, Object, and Goal-Placement.

Table 1: **Target-group adaptation with historical-experience controls.** Means are shown, and Appendix C.1 reports SDs. Overall is task-weighted over 35 tasks. Δ is relative to the fresh structured adapter in pp, and the anchor scores 0.0. **Bold** and underlining mark the top two results.

Method	Spatial (10)	Object (10)	Goal-Pl. (6)	Long-A (3)	Long-B (4)	Long-C (2)	Overall (35)	Δ (pp)
No historical experience								
Full fine-tuning	63.0	<u>83.6</u>	<u>80.0</u>	35.3	24.0	10.0	61.9	+4.0
Fresh LoRA	58.6	72.6	59.3	22.0	27.5	13.0	53.4	−4.5
Fresh structured adapter	68.0	72.6	75.0	18.7	26.5	5.0	57.9	ref.
Pooled historical data								
Pooled-adapter initialization	65.6	72.6	**85.0**	43.3	**62.5**	<u>20.0</u>	<u>66.1</u>	+8.2
Modular historical reuse (R^2-WAM variants)								
Bank composition only	3.0	0.0	0.0	0.0	0.0	0.0	0.9	−57.0
New residual only	39.6	59.0	27.7	5.3	15.0	8.0	35.5	−22.4
Random candidates	<u>70.0</u>	82.0	71.7	<u>48.7</u>	44.0	13.0	65.7	+7.8
R^2-WAM (target-loss candidates)	**78.0**	**85.6**	76.7	**56.7**	<u>50.0</u>	**30.0**	**72.2**	**+14.3**

89.3M trainable parameters, it still reaches 76.2% and 96.0% at 20 and 80, with the latter exceeding an 81.6M parameter-matched LoRA's 90.0%. This points to organization at WAM functional interfaces rather than parameter count. Appendix B.2 gives the full comparison.

RoboCasa365 extends the trend beyond LIBERO. On unseen composite ArrangeBreadBasket, the structured adapter's lead over LoRA grows from 4 to 14 pp between 20 and 80 demonstrations. Its articulated interaction, pick-and-place, and multistage operations extend the trend beyond LIBERO's Spatial-to-Object split.

Overall, the structured adapter is data-efficient and scales beyond LoRA's early plateau. Appendix B.4 gives module ablations before we evaluate further gains from historical reuse.

5.3 Adaptation with Historical Experience

Table 1 uses matched controls for historical experience, support selection, organization, and refinement. Full fine-tuning, fresh LoRA, and fresh structured adaptation use no history. Pooled initialization uses one adapter trained on pooled history. Bank composition only removes the new residual, whereas new residual only removes reuse but retains task-specific exposure. Random candidates and R^2-WAM share the rest of the pipeline and differ only in support selection. Top-4 was selected on a separate LIBERO-90 Development Split before final evaluation, with variants in Appendix C.1.

Historical adapters help, and their selection further matters. Since the anchor scores 0.0% on every target group, all target competence comes from adaptation. Fresh structured adaptation reaches 57.9% and random candidates 65.7%, showing distributed and overlapping capabilities across 15 adapters. Target-loss candidates reach 72.2%, 6.5 pp above random, confirming the value of support aligned with target optimization.

Pooled and modular history organize experience differently. Pooling reaches 66.1%, 8.2 pp above fresh adaptation and close to random reuse at 65.7%, showing that historical data alone forms a useful prior. Target-loss reuse reaches 72.2%, another 6.1 pp. It leads on Spatial, Object, and Long-A, while pooling leads on Goal-Placement, Long-B, and Long-C. Joint training may build cross-task interactions, whereas modular storage preserves separable experience for recombination.

Table 2: **Compact Section 5.4 ablations.** GS and TS denote group-shared and task-specific. Panels summarize personalization, matched-K support construction, and new-residual exposure.

(a) Sharing and personalization

Variant	Sharing			SR
	Cand.	α	η	
Fully GS	GS	GS	GS	57.7
+ TS exposure	GS	GS	TS	58.7
+ TS comp. (GSC)	**GS**	**TS**	**TS**	**72.2**
+ TS candidates	TS	TS	TS	70.6

(b) Historical support

Factor	Configuration				SR
	Method	Evidence	Gran.	K	
Gran.	**GSC**	loss q	route	2	**54.0**
	WQ	loss q	whole	2	43.8
Rank.	**WQ**	loss q	whole	4	**46.3**
	TE	task emb.	whole	4	40.0

(c) New-residual exposure

Exposure	Task 50			Task 9		
	S1	e2	e20	S1	e2	e20
Direct	9	0	9	0	5	9
5-ep ramp	9	8	8	0	4	**10**
Learnable	9	8	**10**	0	**9**	**10**

Historical reuse and new refinement are both necessary. Bank composition only reaches 0.9% under the full target optimization budget, while new residual only reaches 35.5%. Their combination substantially exceeds either component. Task-specific exposure starts at 0.1 and tends to decrease to $\eta_t \approx 0.06$–0.10. The new residual remains essential even at this small exposure.

Learned compositions further reflect target-group heterogeneity. Figure 3(c) shows coefficient dispersion increasing from Object to Spatial to Goal-Placement, with means of 0.711, 1.044, and 1.874. Consistent groups share stable patterns, while heterogeneous tasks weight common candidates differently. These parameter patterns echo the group-wise performance split and motivate Section 6's persistence and consolidation analysis.

5.4 Ablations on Reuse and Refinement

Three ablations test where task-specific freedom is needed, how granularity and ranking evidence shape historical support, and how exposure controls new-residual refinement.

5.4.1 Sharing and Personalization

R^2-WAM combines group-shared and task-specific components within target groups. Starting fully group-shared, we sequentially personalize new-residual exposure, historical composition, and candidate selection to identify task-specific degrees of freedom and measure each marginal effect.

Task-specific composition captures most within-group variation. Starting fully GS at 57.7%, personalizing exposure, composition, and candidates successively yields 58.7%, 72.2%, and 70.6%. The small exposure and large composition gains show that groups can share candidates while tasks need distinct capability combinations. Candidate personalization gives no overall gain but helps heterogeneous groups (Appendix C.1). We therefore use shared candidates with task-specific composition to preserve task-level reuse flexibility within a shared candidate space.

5.4.2 Historical Support Construction

Functional reuse requires deciding both the granularity at which historical adaptations participate in composition and the evidence used to assess their support for current targets. On Long-B, matched-K comparisons alternately hold ranking evidence and composition granularity fixed to isolate each design under the same target setting.

Route-wise composition better captures functional combinations across histories. With target-loss evidence and Top-2 fixed, route-wise GSC reaches 54.0% versus whole-adapter q ranking (WQ) at 43.8%, a 10.2 pp gain. A target's reusable capabilities need not align with one historical adapter because its routes can draw from different histories.

Target-loss evidence better identifies support for optimization than task semantics. At fixed whole-adapter Top-4, target-loss WQ reaches 46.3% versus task-embedding retrieval (TE) at 40.0%, isolating ranking evidence. TE selects H7, H8, H10, and H15, which respectively match Long-B's drawer, fixture-state change, stove, and cabinet manipulations and form a reasonable semantic prior. WQ selects H6, H8, H11, and H12, with less direct matches but higher success. Semantic relevance therefore differs from parameter-level transferability, while q directly identifies support for the target objective (Appendix C.2).

5.4.3 NEW-RESIDUAL EXPOSURE

Different bank coverage means fixed new-residual exposure can disrupt established support or slow learning for low-coverage tasks. We compare direct, fixed-ramp, and learnable exposure on two LIBERO-90 Development Split tasks with different Stage 1 states.

Task-specific learnable exposure adapts refinement to historical coverage. After reuse, Task 50 reaches 9/10, but direct exposure drives it to 0/10 early and disrupts established support. A fixed ramp limits disruption but learns slowly from Task 9's 0/10. Learnable exposure preserves Task 50 and reaches 10/10, while raising Task 9 to 9/10 by refinement epoch 2. Thus, η_t adapts new-residual strength to coverage and balances reuse with refinement.

Together, groups share candidates, task-specific composition handles variation, target-loss evidence builds route-wise support, and task-specific exposure scales new capacity to coverage.

6 DISCUSSION: EXPERIENCE PERSISTENCE AND BANK EXPANSION

R^2-WAM retains a bank-dependent routed state for current targets, then consolidates accumulated experience into the historical bank over a longer deployment timescale.

Short-term persistence retains the routed state or folds it into standalone parameters. After adaptation, $\mathcal{E}^{\text{new}} = \{\phi^*, S^*, \{\boldsymbol{\alpha}_t^*, \eta_t^*\}_{t=1}^M\}$ depends on the bank and is not yet a standalone entry. For one task, $(\boldsymbol{\alpha}^*, \eta^*)$ combines parallel historical and new residuals sharing each input, allowing exact folding into a standalone adapter. A group must first compress its M task-specific pairs into $\bar{\boldsymbol{\alpha}}$ and $\bar{\eta}$, the only approximation. Group-level folding thus suits compositions consistent across tasks as observed in Section 5.3, with both derivations in Appendix E.1–E.2.

Long-term consolidation turns accumulated experience into a new historical adapter. Once related tasks and demonstrations form a stable shared operation structure, we reorganize them as $\mathcal{G}_{N+1}$, train function-structured H_{N+1} from the shared anchor, and update $\mathcal{B} \leftarrow \mathcal{B} \cup \{H_{N+1}\}$. Future groups recompute route-wise transferability and candidates from their demonstrations without retraining a cross-task neural router as the bank grows.

Parameterized experience simplifies bank maintenance. Pooled initialization requires historical demonstrations, whereas R^2-WAM reuses frozen adapters without the original trajectories, reducing raw-data retention. Operation templates directly define bank groups, eliminating learned grouping.

7 RELATED WORKS

Robot policy adaptation. New-task methods use LoRA, structured conditioning, or selective updates (Hu et al., 2021; Kim et al., 2024; Li et al., 2025; Mitra et al., 2025), primitive compositions, action heads, or prompts (Myers et al., 2024; Zhao et al., 2025; Zhang et al., 2026), lightweight test-time updates or memory (Feng et al., 2026; Jiang et al., 2026), and demonstration context (Sridhar et al., 2025; Chen et al., 2025; Zhou et al., 2026). These draw on shared policy priors, while R^2-WAM reuses saved adaptations for later tasks.

Historical parameter reuse. Historical adapters can be retrieved by task semantics or representations (Zhao et al., 2024; Dhasade et al., 2026; Kuzmenko & Shvai, 2026). DMPEL routes over a growing expert library (Lei et al., 2026), while CLARE uses representation novelty for reuse or expansion (Römer et al., 2026). LoRAHub composes frozen modules from few target examples (Huang et al., 2024). R^2-WAM brings target-given reuse to embodied WAMs, selecting histories per route and retaining capacity for missing capabilities.

8 CONCLUSION

We introduce R^2-WAM for history-aware cross-task few-shot WAM adaptation and turn saved adaptations into reusable parameterized experience. It recombines historical residuals across video, context, and action routes and refines capabilities absent from the bank. Experiments show its function-structured space is highly data-efficient and accumulated adaptations outperform fresh adaptation. They therefore form a reusable experience base for future tasks and long-term WAM deployment.

AI USE STATEMENT

OpenAI Codex was used to assist with language editing, presentation and organization, literature search, LaTeX formatting, and revisions to the description and interpretation of results. It was not used to generate experimental data.

REPRODUCIBILITY STATEMENT

Sections 3 and 4.1–4.3 specify the problem formulation, function-structured adapter, target-loss-driven historical selection and composition, and target-group refinement objectives. Section 5.1 describes the data-budget and history-aware evaluation protocols. Appendix A provides the benchmark construction, task splits, shared backbone initialization, training and evaluation settings, deterministic sampling procedure, and hyperparameters; Appendices B–D report baseline selection, parameter-controlled comparisons, complete results, ablations, candidate rankings, sampling stability, system profiling, and historical-adapter analyses. Appendix E gives the derivations and protocols for experience persistence and bank expansion.

REFERENCES

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael 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, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. π_0: A vision-language-action flow model for general robot control, 2026. URL https://arxiv.org/abs/2410.24164.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, Pete Florence, Chuyuan Fu, Montse Gonzalez Arenas, Keerthana Gopalakrishnan, Kehang Han, Karol Hausman, Alexander Herzog, Jasmine Hsu, Brian Ichter, Alex Irpan, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Lisa Lee, Tsang-Wei Edward Lee, Sergey Levine, Yao Lu, Henryk Michalewski, Igor Mordatch, Karl Pertsch, Kanishka Rao, Krista Reymann, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Pierre Sermanet, Jaspiar Singh, Anikait Singh, Radu Soricut, Huong Tran, Vincent Vanhoucke, Quan Vuong, Ayzaan Wahid, Stefan Welker, Paul Wohlhart, Jialin Wu, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023. URL https://arxiv.org/abs/2307.15818.

Guangyan Chen, Meiling Wang, Qi Shao, Zichen Zhou, Weixin Mao, Te Cui, Minzhao Zhu, Yinan Deng, Luojie Yang, Zhanqi Zhang, Yi Yang, Hua Chen, and Yufeng Yue. See once, then act: Vision-language-action model with task learning from one-shot video demonstrations, 2025. URL https://arxiv.org/abs/2512.07582.

Akash Dhasade, Anne-Marie Kermarrec, Igor Pavlovic, Diana Petrescu, Rafael Pires, Mathis Randl, and Martijn de Vos. Effective lora adapter routing using task representations, 2026. URL https://arxiv.org/abs/2601.21795.

Yusen Feng, Bingchen Han, Jiangran Lyu, Kai Liu, Yixin Zheng, Yuxuan Wan, Weiheng Liu, Sun Han, Ruiqin Li, Yulong Zhang, Fangfu Liu, Xuesong Shi, Libin Liu, Yizhou Wang, Zhizheng Zhang, and He Wang. Wam-ttt: Steering world-action models by watching human play at test time, 2026. URL https://arxiv.org/abs/2607.06988.

Generalist Team. Gen-1.5: Embodied foundation models are one-shot learners. *Generalist AI Blog*, 2026. https://generalistai.com/blog/gen-1.5.

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, 2021. URL https://arxiv.org/abs/2106.09685.

Chengsong Huang, Qian Liu, Bill Yuchen Lin, Tianyu Pang, Chao Du, and Min Lin. Lorahub: Efficient cross-task generalization via dynamic lora composition, 2024. URL https://arxiv.org/abs/2307.13269.

Yunfan Jiang, Yevgen Chebotar, Ruijie Zheng, Fengyuan Hu, Yunhao Ge, Jimmy Wu, Tianyuan Dai, Scott Reed, Li Fei-Fei, Yuke Zhu, and Linxi "Jim" Fan. Robottt: Context scaling for robot policies, 2026. URL https://arxiv.org/abs/2607.15275.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag 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, 2024. URL https://arxiv.org/abs/2406.09246.

Dmytro Kuzmenko and Nadiya Shvai. Moira: Modular instruction routing architecture for multi-task robotics, 2026. URL https://arxiv.org/abs/2507.01843.

Yuheng Lei, Sitong Mao, Shunbo Zhou, Hongyuan Zhang, Xuelong Li, and Ping Luo. Dynamic mixture of progressive parameter-efficient expert library for lifelong robot learning, 2026. URL https://arxiv.org/abs/2506.05985.

Lin Li, Qihang Zhang, Yiming Luo, Shuai Yang, Ruilin Wang, Fei Han, Mingrui Yu, Zelin Gao, Nan Xue, Xing Zhu, Yujun Shen, and Yinghao Xu. Causal world modeling for robot control, 2026. URL https://arxiv.org/abs/2601.21998.

Puhao Li, Yingying Wu, Ziheng Xi, Wanlin Li, Yuzhe Huang, Zhiyuan Zhang, Yinghan Chen, Jianan Wang, Song-Chun Zhu, Tengyu Liu, and Siyuan Huang. Controlvla: Few-shot object-centric adaptation for pre-trained vision-language-action models, 2025. URL https://arxiv.org/abs/2506.16211.

Yuankai Luo, Woping Chen, Tong Liang, and Zhenguo Li. Coral: Scalable multi-task robot learning via lora experts, 2026. URL https://arxiv.org/abs/2603.09298.

Chancharik Mitra, Yusen Luo, Raj Saravanan, Dantong Niu, Anirudh Pai, Jesse Thomason, Trevor Darrell, Abrar Anwar, Deva Ramanan, and Roei Herzig. Mechanistic finetuning of vision-language-action models via few-shot demonstrations, 2025. URL https://arxiv.org/abs/2511.22697.

Vivek Myers, Bill Chunyuan Zheng, Oier Mees, Sergey Levine, and Kuan Fang. Policy adaptation via language optimization: Decomposing tasks for few-shot imitation, 2024. URL https://arxiv.org/abs/2408.16228.

NVIDIA, :, Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi "Jim" Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyu Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhenjia Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. Gr00t n1: An open foundation model for generalist humanoid robots, 2025. URL https://arxiv.org/abs/2503.14734.

Ralf Römer, Yi Zhang, Yuming Li, and Angela P. Schoellig. Clare: Continual learning for vision-language-action models via autonomous adapter routing and expansion, 2026. URL https://arxiv.org/abs/2601.09512.

Skild AI. Introducing s1: In-context learning for robotics, August 2026. URL https://skild.ai/blogs/s1.

Kaustubh Sridhar, Souradeep Dutta, Dinesh Jayaraman, and Insup Lee. Ricl: Adding in-context adaptability to pre-trained vision-language-action models, 2025. URL https://arxiv.org/abs/2508.02062.

Angen Ye, Boyuan Wang, Chaojun Ni, Guan Huang, Guosheng Zhao, Hao Li, Hengtao Li, Jie Li, Jindi Lv, Jingyu Liu, Min Cao, Peng Li, Qiuping Deng, Wenjun Mei, Xiaofeng Wang, Xinze Chen, Xinyu Zhou, Yang Wang, Yifan Chang, Yifan Li, Yukun Zhou, Yun Ye, Zhichao Liu, and Zheng Zhu. Gigaworld-policy: An efficient action-centered world–action model, 2026a. URL https://arxiv.org/abs/2603.17240.

Seonghyeon Ye, Yunhao Ge, Kaiyuan Zheng, Shenyuan Gao, Sihyun Yu, George Kurian, Suneel Indupuru, You Liang Tan, Chuning Zhu, Jiannan Xiang, Ayaan Malik, Kyungmin Lee, William Liang, Nadun Ranawaka, Jiasheng Gu, Yinzhen Xu, Guanzhi Wang, Fengyuan Hu, Avnish Narayan, Johan Bjorck, Jing Wang, Gwanghyun Kim, Dantong Niu, Ruijie Zheng, Yuqi Xie, Jimmy Wu, Qi Wang, Ryan Julian, Danfei Xu, Yilun Du, Yevgen Chebotar, Scott Reed, Jan Kautz, Yuke Zhu, Linxi "Jim" Fan, and Joel Jang. World action models are zero-shot policies, 2026b. URL https://arxiv.org/abs/2602.15922.

Tianyuan Yuan, Zibin Dong, Yicheng Liu, and Hang Zhao. Fast-wam: Do world action models need test-time future imagination?, 2026. URL https://arxiv.org/abs/2603.16666.

Wenbo Zhang, Jianxiong Li, Shuai Yang, Sijin Chen, Jiajun Liu, Lingqiao Liu, and Xiao Ma. Ttt-vla: Test-time latent prompt optimization for vision-language-action models, 2026. URL https://arxiv.org/abs/2606.03127.

Ruihan Zhao, Tyler Ingebrand, Sandeep Chinchali, and Ufuk Topcu. Mos-vla: A vision-language-action model with one-shot skill adaptation, 2025. URL https://arxiv.org/abs/2510.16617.

Ziyu Zhao, Leilei Gan, Guoyin Wang, Yuwei Hu, Tao Shen, Hongxia Yang, Kun Kuang, and Fei Wu. Retrieval-augmented mixture of lora experts for uploadable machine learning, 2024. URL https://arxiv.org/abs/2406.16989.

Jiaming Zhou, Qihang Zhang, Gangwei Xu, Cunxin Fan, Yujie Zhao, Ruilin Wang, Yiming Luo, Shuai Yang, Xing Zhu, Yujun Shen, Junwei Liang, and Yinghao Xu. Zero-wam: In-context world-action modeling from human videos for open-ended task generalization, 2026. URL https://arxiv.org/abs/2608.26103.

A History-Aware Benchmark and Experimental Protocol

We detail the benchmark, data splits, training, and evaluation protocols for Section 5.1.

A.1 Benchmark Construction and Task Splits

Robot onboarding. The anchor policy uses 500 demonstrations balanced across the 10 manipulation families below. This split covers varied manipulation primitives and interaction dynamics without extending the anchor to directly cover extensive object-specific semantics or multi-stage task compositions.

Table 3: **Robot onboarding data.** The 500 demonstrations evenly cover 10 manipulation families. We list task IDs, demonstration counts, and covered primitives or interactions.

Capability family	IDs	Demos	Covered primitive / interaction
Drawer open / close	7, 11, 0, 28	50	prismatic pull / push
Microwave open / close	35, 33	50	revolute articulation
Stove on / off	20, 44, 39	50	rotary / state actuation
Bowl → drawer	2, 29	50	articulated place-in
Soup → tray	55	50	generic place-in
Pan → stove	18	50	large-object place-on
Mug → plate	71	50	small-object place-on
Bowl → bowl	17	50	stacking / precision contact
Bowl → right of plate	37	50	relative free-space placement
Pan → under shelf	42	50	constrained placement

Historical adapter bank. The historical bank is organized into 15 task groups by reusable object–target relations, spatial relations, and operation structures, rather than original LIBERO suite boundaries. Each group corresponds to an independently trained and frozen function-structured adapter.

Table 4: **Complete historical adapter bank task split.** Each of H1–H15 corresponds to one task group. We list capability families, LIBERO-90 task IDs, and primary reusable operation structures.

Adapter	Capability family	IDs	Main reusable structure
H1	Food → Basket	46, 48, 52, 53	same receptacle across food instances
H2	Food → Tray	56, 57, 58, 59	tray receptacle placement
H3	Spatial Bowl → Plate	12, 13, 14	source-side spatial disambiguation
H4	Relative Free-space Placement	34, 69, 70	reference-relative target region
H5	Book → Caddy Compartments	73, 74, 75, 78, 79, 80	compartment-specific placement
H6	Spatial Book → Shelf	86, 87, 88, 89	source disambiguation + shelf geometry
H7	Drawer Place + Close	3, 4, 5	placement + fixture-state composition
H8	Object → Drawer	24, 26, 32	atomic drawer placement
H9	Moka → Stove	19, 38	atomic moka/stove placement
H10	Stove–Pan Composition	21, 45	stove/pan multi-step structure
H11	Pan Support Placement	40, 41	pan support geometry
H12	Mug → Spatial Plate	65, 66, 67, 68, 72	mug/plate spatial relations
H13	Object → Right of Caddy	76, 84, 85	caddy-relative placement
H14	Stack / Stack-and-Place	16, 63, 64	stacking and compositional placement
H15	Bowl → Cabinet Top	10, 15, 25, 31, 43	cabinet-top placement

Development split. The Development Split is disjoint from the anchor and historical bank and is reserved for method development and hyperparameter selection.

Table 5: **Development Split.** Five task families support method development and hyperparameter selection and are separated from the anchor, historical bank, and final target evaluation.

Dev family	IDs	Relation to historical bank
D1 Basket Family Expansion	47, 49, 50, 51, 54	H1-related held-out objects / scenes
D2 Tray Family Expansion	60, 61, 62	H2-related placement with object/spatial variation
D3 Caddy New-Scene Transfer	81, 82, 83	H5-related new-scene transfer
D4 Drawer / Fixture Workflow	1, 6, 22, 23	articulation and fixture-state composition
D5 Bowl → Plate Transfer	9, 30, 36	H3-related bowl-to-plate transfer

Target-group reorganization. LIBERO-Spatial and LIBERO-Object retain their original suite organization because their tasks share consistent structures, whereas the more heterogeneous Goal and Long-Horizon suites are reorganized by shared operations. We define Goal-Placement as a controlled set with the common single-step *put A on B* structure; Goal-3, *open the top drawer and put the bowl inside*, instead joins Long-B because its composite manipulation and fixture-state transition match that group. We exclude the five tasks in Table 6: Goal-0, Goal-5, and Goal-7 introduce fixture opening, planar pushing, or fixture activation rather than the intended placement structure, while Long-5 and Long-8 do not align cleanly with the shared operational skeletons of Long-A/B/C. These exclusions preserve within-group comparability and prevent a few structurally distinct tasks from dominating group-level results.

Table 6: **Tasks excluded from the reorganized target groups.** The Goal exclusions keep Goal-Placement in a common *put A on B* form; the Long exclusions avoid singleton operation structures within the Long-Horizon groups.

Task	Instruction	Exclusion reason
Goal-0	*open the middle drawer of the cabinet*	Not *put A on B*: fixture opening.
Goal-5	*push the plate to the front of the stove*	Not *put A on B*: planar pushing.
Goal-7	*turn on the stove*	Not *put A on B*: fixture activation.
Long-5	*pick up the book and place it in the back compartment of the caddy*	Atomic; no shared Long structure.
Long-8	*put both moka pots on the stove*	Different dynamics from Long-A/B/C.

Table 7: **Group-level reorganization of held-out targets.** Groups follow source and shared operation structure; Goal and Long-A denote Goal-Placement and Long-A-core.

Target group	Included tasks	Shared structure
Spatial	original LIBERO-Spatial tasks	spatially conditioned bowl-to-plate manipulation
Object	original LIBERO-Object tasks	object-conditioned basket placement
Goal	Goal-1, 2, 4, 6, 8, 9	single-goal object / receptacle manipulation
Long-A	Long-0, 1, 7	two objects → same target (basket)
Long-B	Long-2, 3, 9 + Goal-3	manipulation + fixture state change
Long-C	Long-4, 6	two distinct subgoals with spatial / target relations

Leakage control. We exclude the following LIBERO-90 tasks because their semantics or structures nearly duplicate final evaluation tasks.

Table 8: **LIBERO-90 tasks excluded to prevent leakage.** Their semantics or operation structures duplicate evaluation tasks, so they are omitted from the anchor, bank, and Development Split.

ID	Instruction	Exclusion reason
8	*open the top drawer of the cabinet and put the bowl in it*	nearly equivalent to Goal-3
27	*put the wine bottle on the wine rack*	nearly equivalent to Goal-9
77	*pick up the book and place it in the back compartment of the caddy*	identical / near-identical to Long-5

A.2 Training and Evaluation Details

This section describes the shared FastWAM implementation, data-budget evaluation, history-aware main evaluation, and Development Split ablations used only for mechanism analysis.

Table 9: **Shared FastWAM backbone initialization for both evaluation protocols.**

Component	Setting
video DiT	initialized from Wan2.2-TI2V-5B
Text encoder	Wan2.1-T2V-1.3B text encoder
action DiT	initialized by interpolation from the video DiT
Main-experiment seeds	42, 43, 44

Data-budget evaluation protocol. We report the data budgets, optimization, checkpoints, and closed-loop evaluation used for the LIBERO-Object and RoboCasa365 experiments in Section 5.2.

Table 10: **Data-budget training and evaluation settings.** LR and BS denote learning rate and global batch size. L/RC denotes LIBERO/RoboCasa365. LIBERO uses 20/40/80/160/320/454 demonstrations, with BS 64 through 80 and 128 thereafter. RoboCasa365 uses 20/80 demonstrations and BS 256. Every adaptation runs for 10 epochs. For LoRA and the structured adapter, the table gives the 20-demonstration reference LR, and the remaining LRs are rescaled by optimizer-step count. Each adapted model uses 50 rollouts.

Method	Budget	LR / schedule	BS (L / RC)	Loss	Checkpoint
Anchor training	10 epochs	1×10^{-4}	128 / 512	video + action	final
Full FT	10 epochs	1×10^{-4}, fixed	64/128 / 256	action	best loss
LoRA	10 epochs	ref. 3×10^{-3}	64/128 / 256	action	best loss
Structured adapter	10 epochs	ref. 1×10^{-3}	64/128 / 256	action	best loss

History-aware evaluation protocol. We give the history-aware protocol for Sections 5.3–5.4, covering bank preparation, q-based ranking, both adaptation stages, and closed-loop evaluation.

Table 11: **History-aware training and evaluation settings.** LR and BS denote learning rate and global batch size. Bank LRs vary by group. The three fresh baselines and Stage 2 use the same valid fit3 windows. Stage 2 uses LRs 10^{-3}, 10^{-4}, and 10^{-2} for ϕ and the input encoders, α, and η, respectively, with cosine decay.

Stage / item	Training data	Budget	LR / schedule	BS	Loss
Anchor	500 onboarding demos (A.1)	10 epochs	1×10^{-4}	128	video + action
Bank H1–H15	groups (A.1)	10 epochs	$(2.5\text{–}3.5) \times 10^{-4}$	64	action
Full FT	all valid fit3 windows	20 epochs	1×10^{-4}	64	action
Fresh LoRA	all valid fit3 windows	20 epochs	1×10^{-3}	64	action
Fresh structured adapter	all valid fit3 windows	20 epochs	1×10^{-3}	64	action
Stage 0: q	24 windows $\times$ 3 NoisePacks	—	—	—	action
Stage 1: reuse	same 72 instances as q	108 steps	0.05, fixed	64	action
Stage 2: refinement	all valid fit3 windows	20 epochs	cosine decay	64	action

Deterministic q support. We divide each target trajectory into 8 approximately equal temporal bins and use seeded-uniform sampling to fix one window per bin. Thus each task's q support contains

$$3 \text{ trajectories} \times 8 \text{ bins} \times 3 \text{ NoisePacks} = 72 \text{ window–NoisePack instances} \quad (24 \text{ unique windows}). \tag{12}$$

Each sampled window uses 3 fixed, paired NoisePacks. The q computation and Stage 1 reuse the same windows, noise, and timesteps, while Stage 2 uses all valid fit3 windows. Appendix C.3 tests sensitivity to NoisePack and temporal-window sampling.

B DATA-BUDGET ADAPTATION

This appendix supplements Section 5.2 with baseline selection, parameter-controlled comparisons, complete data-budget results, and module-wise ablations.

B.1 LORA BASELINE SELECTION

Targeted rank sweep. LoRA uses separate ranks and α values for the action and video branches. Rather than exhaustively searching the two-dimensional grid, we test compact, video-heavy, action-heavy, and substantially larger configurations. Each branch uses α equal to its rank, so the sweep compares complete branch-capacity allocations rather than isolating rank from α. We order the trials to expose allocation trends and select the baseline by closed-loop success.

Table 12: **Targeted rank sweep for the LoRA baseline.** We compare branch ranks, trainable parameters, and success rate (SR) over 500 LIBERO-Object rollouts. Bold marks the selected setting.

Action r	**Video r**	**Action α**	**Video α**	**Params**	**Successes**	**SR**
16	4	16	4	30.772M	437 / 500	87.4%
32	**8**	**32**	**8**	**61.492M**	**460 / 500**	**92.0%**
32	16	32	16	81.645M	450 / 500	90.0%
32	32	32	32	121.949M	435 / 500	87.0%
32	64	32	64	202.558M	432 / 500	86.4%
64	16	64	16	122.932M	434 / 500	86.8%
8	32	8	32	90.983M	428 / 500	85.6%
160	40	160	40	307.252M	429 / 500	85.8%

Baseline selection. Increasing either branch rank in isolation does not improve the baseline. With action rank fixed at 32 and LR at 2.5×10^{-4}, raising video rank from 16 to 32 and 64 reduces SR from 90.0% to 87.0% and 86.4%. With video rank fixed at 16, raising action rank from 32 to 64 similarly lowers SR from 90.0% to 86.8%, showing that simply increasing action-side rank also fails to improve performance. The video-heavy $(8, 32)$ and compact $(16, 4)$ settings reach only 85.6% and 87.4%, so neither skewed allocation nor reduced capacity surpasses the selected configuration. Even $(160, 40)$ uses 307.252M parameters but reaches only 85.8%, confirming that total parameter count is insufficient. We therefore select $(32, 8)$ with LR 3.5×10^{-4}, which achieves the highest completed-trial SR at 92.0%.

B.2 PARAMETER-CONTROLLED STRUCTURED ADAPTER

The full function-structured adapter has approximately 342M trainable parameters. For capacity control, we use a shared full-rank base and an independent rank-128 residual at each layer, reducing the count to 89.256M. All functional insertion points and route organization remain unchanged. Compressing only the cross-layer representation lets Section 5.2 separate structured adaptation from simply adding parameters.

Table 13: **Parameter-capacity comparison of structured adapters and LoRA.** High-capacity and approximately 90M matched settings report trainable parameters and success over 500 rollouts.

Regime	**Method**	**Params**	**Action r**	**Video r**	**Successes**	**SR**
High-cap. (∼300M)	Structured Adapter	**342.0M**	—	—	486 / 500	**97.2%**
	LoRA	307.252M	160	40	429 / 500	85.8%
Matched (∼90M)	Compressed Structured Adapter	**89.256M**	—	—	480 / 500	**96.0%**
	LoRA	81.645M	32	16	450 / 500	90.0%

B.3 Complete Data-Budget Results

Section 5.2 compares methods from the same spatial anchor. We add full fine-tuning without the anchor, with and without G_v, to isolate initialization and adaptation structure across data budgets.

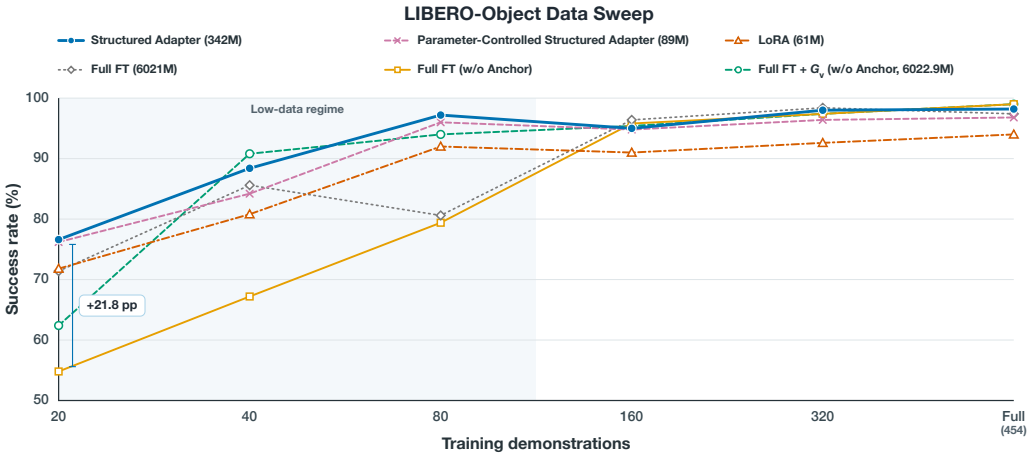


Figure 4: **Complete LIBERO-Object data-budget sweep.** Across 20–454 demonstrations, four settings use the spatial anchor and two Full Fine-Tuning variants omit it, one with G_v. Shading marks the 20–80 range.

Table 14: **Complete LIBERO-Object data-budget sweep.** Six settings report initialization, trainable parameters, and success (%). Spatial denotes the anchor, none its omission, and Full 454 demonstrations.

Method	Init.	Params	20	40	80	160	320	Full (454)
Structured Adapter	Spatial	342M	**76.6**	**88.4**	**97.2**	95.0	98.0	98.2
Compressed Structured Adapter	Spatial	89.3M	76.2	84.2	96.0	94.8	96.4	96.8
LoRA	Spatial	61.5M	71.8	80.8	92.0	91.0	92.6	94.0
Full Fine-Tuning	Spatial	6020.7M	71.4	85.6	80.6	**96.4**	**98.4**	97.4
Full Fine-Tuning	none	6020.7M	54.8	67.2	79.4	95.8	97.4	**99.0**
Full Fine-Tuning + G_v	none	6022.9M	62.4	**90.8**	94.0	95.4	97.4	**99.0**

1. Anchor initialization.

1) The spatial anchor improves the low-data starting point of full fine-tuning. We compare identical parameter spaces with and without spatial-anchor initialization. The anchor gains 16.6 pp and 18.4 pp with 20 and 40 demonstrations, while the curves converge at larger budgets. It therefore improves the adaptation starting point and low-data sample efficiency rather than the data-rich ceiling. The comparison in Section 5.2 includes this initialization benefit for full fine-tuning.

Table 15: **Effect of the spatial anchor on Full Fine-Tuning.** Closed-loop success rates (%) across budgets. The anchor gain is the difference (pp) between using and omitting the spatial anchor.

Method	20	40	80	160	320	Full (454)
Full FT w/ spatial anchor	71.4	85.6	80.6	96.4	98.4	97.4
Full FT w/o spatial anchor	54.8	67.2	79.4	95.8	97.4	99.0
Anchor gain	**+16.6**	**+18.4**	+1.2	+0.6	+1.0	−1.6

2) Joint WAM–adapter pretraining does not provide a better cross-task initialization. We jointly train the WAM and full structured adapter on Spatial, then initialize Object adaptation from both sets of weights. During target adaptation, the WAM backbone is frozen and only the adapter and corresponding input-side encoders are trained. The control uses the same spatial anchor with the identity or zero-effect adapter initialization from Section 4.1. Joint pretraining falls below the control by 4.2 pp with 20 demonstrations and 3.0 pp with 80. Prelearning adapter parameters therefore does not inherently improve the target-task starting point. Identity or zero-effect initialization instead leaves the anchor policy unchanged initially and preserves a clear anchor-relative residual interpretation throughout adaptation.

Table 16: **Structured adapter initialization comparison.** Identity or zero-effect initialization is compared with joint WAM–adapter pretraining at 20 and 80 demonstrations. Difference is the latter minus the former.

Initialization	**20 demos**	**80 demos**
Spatial anchor + identity / zero-effect adapter	**76.6%**	**97.2%**
Jointly pretrained Spatial WAM + adapter	72.4%	94.2%
Difference (jointly pretrained − control)	−4.2 pp	−3.0 pp

2. Video-to-action value gate.

The video-to-action value gate improves full fine-tuning when target data are limited. We add G_v to full fine-tuning without the spatial anchor, isolating its modulation of video values read by action queries. It improves success by 7.6 pp, 23.6 pp, and 14.6 pp with 20, 40, and 80 demonstrations, with the largest gain at 40. The advantage disappears once full fine-tuning approaches saturation from 160 demonstrations onward. Thus, action-query-specific video-value modulation remains effective even when the full backbone is trainable. Its explicit interface improves low-data optimization, while sufficient data allow the original parameters to learn similar adjustments and reduce the gate's additional benefit.

Table 17: **Independent contribution of the video-to-action value gate.** Full Fine-Tuning with and without G_v, both without the spatial anchor. Success rates are in % and G_v gains in pp.

Method	**20**	**40**	**80**	**160**	**320**	**Full (454)**
Full FT w/o anchor	54.8	67.2	79.4	95.8	97.4	99.0
Full FT + G_v w/o anchor	62.4	90.8	94.0	95.4	97.4	99.0
G_v **gain**	**+7.6**	**+23.6**	**+14.6**	−0.4	0.0	0.0

3. PEFT scaling.

Structured adaptation continues to scale after LoRA reaches an early plateau. LoRA rises from 71.8% with 20 demonstrations to 92.0% with 80, then remains around 91.0–94.0%. The full structured adapter reaches 97.2% with 80 and stays near saturation. Its 89.3M compressed variant still exceeds LoRA by 2.8–4.4 pp at every budget. Together with the approximately 90M comparison in Appendix B.2, this shows that parameter count alone cannot explain the scaling gap.

Table 18: **Data-budget scaling of compressed structured adaptation and LoRA.** We report closed-loop success across budgets and the compressed adapter's gain over LoRA in pp.

Method	**20**	**40**	**80**	**160**	**320**	**Full (454)**
Compressed Structured Adapter (89.3M)	76.2	84.2	96.0	94.8	96.4	96.8
LoRA (61.5M)	71.8	80.8	92.0	91.0	92.6	94.0
Difference	**+4.4**	**+3.4**	**+4.0**	**+3.8**	**+3.8**	**+2.8**

B.4 MODULE-WISE ABLATION

We conduct module-wise ablations with 80 demonstrations, using the same untuned learning rate of 10^{-4} for every configuration. Section 5.2 and Appendix B.3 use tuned settings, so the analysis here uses matched differences within this ablation. Starting from input-side encoders, we progressively add F_a and O_a on the action route, G_v and B_v on the video route, and G_c and O_c on the context route. Every configuration is trained independently from the same spatial anchor rather than continued from the preceding checkpoint, so each increment compares matched structural expansions.

The action and video routes establish target behavior jointly. Input-side alignment alone yields 0.0% success. Adding F_a and O_a raises it to 19.8% and 30.2%, progressively establishing adaptation along the action path. Adding G_v then raises success to 72.0%, and B_v further increases it to 88.0%. Since G_v does not enable success when added directly to the input-only setting, its gain depends on an existing action path. Effective adaptation therefore requires both action transformation and world-information adaptation.

Table 19: **Module-wise ablation of the function-structured adapter.** All configurations are trained independently from the same spatial anchor with 80 demonstrations and an untuned learning rate of 10^{-4}. Check marks indicate enabled modules, and Δ SR is the change from the preceding row (pp).

Configuration	F_a	O_a	G_v	B_v	G_c	O_c	**Params**	**SR**	**Δ SR**
Input encoders only							0.045M	0.0%	—
+ action FFN	✓						16M	19.8%	+19.8 pp
+ action attn-out	✓	✓					24M	30.2%	+10.4 pp
+ V2A value gate	✓	✓	✓				26M	72.0%	+41.8 pp
+ video FFN beta	✓	✓	✓	✓			309M	88.0%	+16.0 pp
+ context value gate	✓	✓	✓	✓	✓		334M	83.8%	−4.2 pp
+ context readout	✓	✓	✓	✓	✓	✓	342M	**93.4%**	**+9.6 pp**

The context route requires coordinated value selection and readout. On the four-module action-and-video base at 88.0%, G_c alone lowers success to 83.8% and O_c alone gives a limited gain to 89.6%. Their combination reaches 93.4%, the highest result. The two modules therefore form complementary interfaces. Once G_c changes context value selection, the action-side readout O_c must adapt to the resulting representation.

Table 20: **Composition ablation for context modules.** With $G_v + B_v + F_a + O_a$ fixed, we add G_c, O_c, or both. SR is closed-loop success rate with 80 demonstrations.

Context variant	**SR**
Base only	88.0%
Base $+G_c$	83.8%
Base $+O_c$	89.6%
Base $+G_c + O_c$	**93.4%**

At the route level, performance progresses as input only (0.0%) $\rightarrow$ action (30.2%) $\rightarrow$ action+video (88.0%) $\rightarrow$ action+video+context (93.4%). The three functional routes thus provide complementary adaptation degrees of freedom.

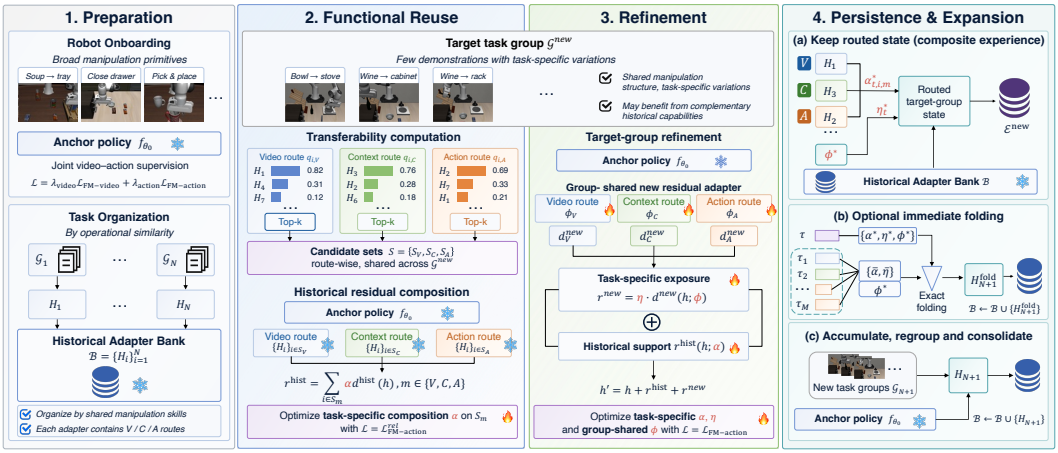


Figure 5: **Detailed R^2-WAM deployment lifecycle.** Task-group adapters build the historical bank from the anchor. For few-shot targets, R^2-WAM recombines bank residuals across video, context, and action routes, learns missing capabilities, and supports folding or group-level consolidation.

C History-Aware Adaptation Analysis

This appendix complements Sections 5.3–5.4 with fixed candidate-budget variants omitted from the main table, a single-task RoboCasa365 routing result, complete historical candidate rankings, q-sampling stability, and per-task adaptation costs. Section 5.4.3 already reports the full new-residual exposure comparison.

C.1 Complete History-Aware Results

Table 21 adds all controls and fixed Top-2 and Top-4 variants to Section 5.3. FGS shares candidates, composition, and exposure; TSE makes only exposure task-specific. GSC also makes composition task-specific, and TSC makes all three task-specific. For seeds 42–44, each subtask uses 50 rollouts. We report the mean and sample standard deviation, with task-weighted Overall over 35 tasks.

Table 21: **Complete history-aware results and controls.** Mean ± sample SD (%) over seeds 42–44 with 50 rollouts per subtask. Overall is task-weighted.

Method	Spatial (10)	Object (10)	Goal-Pl. (6)	Long-A (3)	Long-B (4)	Long-C (2)	Overall (35)
No historical experience							
Anchor	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Full fine-tuning	63.0 ± 0.5	83.6 ± 0.5	80.0 ± 0.7	35.3 ± 1.3	24.0 ± 1.5	10.0 ± 0.0	61.9 ± 0.4
Fresh LoRA	58.6 ± 0.2	72.6 ± 0.4	59.3 ± 0.6	22.0 ± 1.8	27.5 ± 0.9	13.0 ± 1.0	53.4 ± 0.4
Fresh structured adapter	68.0 ± 0.5	72.6 ± 0.2	75.0 ± 0.3	18.7 ± 0.0	26.5 ± 1.3	5.0 ± 1.0	57.9 ± 0.2
Pooled historical data							
Pooled-adapter init.	65.6 ± 0.4	72.6 ± 0.6	**85.0 ± 0.9**	43.3 ± 1.2	**62.5 ± 0.5**	20.0 ± 1.0	66.1 ± 0.2
Modular historical reuse (R^2-WAM variants)							
New residual only	39.6 ± 0.6	59.0 ± 0.3	27.7 ± 0.6	5.3 ± 1.8	15.0 ± 1.0	8.0 ± 1.0	35.5 ± 0.2
Bank composition only	3.0 ± 0.2	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.9 ± 0.1
Random candidates	70.0 ± 0.5	82.0 ± 0.3	71.7 ± 0.6	48.7 ± 1.3	44.0 ± 1.0	13.0 ± 1.7	65.7 ± 0.2
R^2-WAM (FGS)	60.0 ± 0.5	81.6 ± 0.5	44.3 ± 0.7	52.0 ± 0.7	36.5 ± 1.5	18.0 ± 1.0	57.7 ± 0.3
R^2-WAM (TSE)	61.6 ± 0.4	85.0 ± 0.0	44.3 ± 1.0	46.7 ± 1.2	34.0 ± 0.5	23.0 ± 1.0	58.7 ± 0.2
R^2-WAM (GSC, Top-2)	72.6 ± 0.4	87.0 ± 0.5	71.7 ± 0.7	40.0 ± 1.8	54.0 ± 1.3	5.0 ± 1.7	67.8 ± 0.2
R^2-WAM (GSC, Top-4)	**78.0 ± 0.2**	85.6 ± 0.5	76.7 ± 0.0	56.7 ± 0.7	50.0 ± 1.3	**30.0 ± 1.7**	**72.2 ± 0.5**
R^2-WAM (TSC, Top-2)	76.0 ± 0.4	**89.0 ± 0.3**	**87.7 ± 0.3**	48.7 ± 1.8	54.0 ± 1.0	3.0 ± 1.0	**72.7 ± 0.3**
R^2-WAM (TSC, Top-4)	74.6 ± 0.5	78.0 ± 0.5	84.3 ± 0.3	**58.7 ± 1.8**	**59.0 ± 0.5**	13.0 ± 1.0	70.6 ± 0.3

Candidate width and personalization favor different groups. Relative to Top-2, GSC Top-4 gains 4.4 pp Overall through Long-A and Long-C gains of 16.7 and 25.0 pp, despite lower Object and Long-B success. TSC Top-4 gains 10.0, 5.0, and 10.0 pp on Long-A, Long-B, and Long-C, but

loses 11.0 pp on Object and 3.4 pp on Goal-Placement, leaving Top-2 best Overall at 72.7%. At Top-2, TSC raises Goal-Placement, Long-A, and Overall by 16.0, 8.7, and 4.9 pp over GSC.

Route-wise historical support also improves the RoboCasa365 target. We evaluate the single held-out ArrangeBreadBasket task with 20 demonstrations and a bank of six frozen task adapters trained from an anchor pretrained on pretrain-composite and pretrain-atomic data. GSC ranks the bank separately on the video, context, and action routes using a fixed five-demonstration subset of the same target data, selects Top-2 per route, and refines the composition coefficients, new-residual exposure, and fresh residual on all 20 demonstrations. The single-task group thus has one route-wise composition and exposure state. At 62.0%, the action-loss-best checkpoint exceeds the prior fresh adapter and LoRA references by 10.0 and 14.0 pp.

Table 22: **RoboCasa365 single-task routing with 20 demonstrations.** All methods adapt to ArrangeBreadBasket and use 50 closed-loop episodes. The historical bank contains six frozen task adapters. Baseline rows are previously reported reference runs.

Method	Historical support	Demos	SR (%)	Δ Fresh (pp)
Fresh structured adapter	None	20	52.0	ref.
LoRA	None	20	48.0	−4.0
R^2-WAM (GSC Top-2)	6-adapter bank	20	**62.0**	+10.0

C.2 Ordered Candidate Rankings

Section 5.4.2 compares historical support construction on Long-B. Table 23 gives each complete descending ranking; Top-K takes its first K entries, avoiding repeated nested sets.

GSC represents support unavailable to a shared whole-adapter ranking. It ranks routes separately, whereas WQ and TE share one ranking. GSC Top-2 yields H2, H5, H6, H9, and H12; WQ Top-6 yields H6, H11, H12, H8, H2, and H1. Each list deterministically defines every Top-K set.

Table 23: **Historical candidate rankings on Long-B.** Top-K takes each list's first K entries. GSC ranks routes independently; WQ and TE share one ranking.

Method	Organization	Route	Ordered candidates, highest to lowest
GSC	Route-wise	video	H6 → H9 → H8 → H1
GSC	Route-wise	context	H12 → H2 → H6 → H7
GSC	Route-wise	action	H6 → H5 → H8 → H9
WQ	Shared V/C/A	all	H6 → H11 → H12 → H8 → H2 → H1 → H10 → H9
TE	Shared V/C/A	all	H15 → H8 → H10 → H7

C.3 Sampling Stability of Route-Wise Transferability

We examine whether target-loss-driven transferability is sensitive to the stochastic support used to estimate q. On LIBERO-90 Development Split Task 9, we vary either the NoisePack or the temporal-stratified window manifest while keeping the downstream Stage 1 and Stage 2 configurations fixed. Each manifest has 24 windows, one uniformly sampled from each of eight temporal bins in three demonstrations. W0/all is the reference condition. NP0–NP2 use one of the three fixed NoisePacks individually on W0, while W1/all and W2/all change only the window manifest. In every condition, q ranks H1–H15 independently on the video, context, and action routes before Top-4 selection.

Top-4 support is more stable than the complete ranking. Across NoisePack pairs, mean route-wise Top-4 overlap is 75.0–83.3%, while Kendall's τ_b over the full H1–H15 rankings is 0.511–0.714. Changing the window manifest gives the same 75.0–83.3% overlap, with τ_b of 0.683–0.765. The most sensitive individual comparison is NP0 versus NP1 on the video route, where three of four Top-4 candidates remain shared despite $\tau_b = 0.219$. Thus, changes in the complete ordering do not necessarily replace most of the candidate support used for composition.

Table 24: **Sampling stability of target-loss-driven transferability on LIBERO-90 Development Split Task 9.** Top-4 overlap and Kendall's τ_b are averaged across video, context, and action routes. Kendall's τ_b is computed on the complete H1–H15 ranking.

(a) Candidate and ranking stability

Pair	Top-4 overlap	Kendall τ_b
NoisePack		
NP0–NP1	75.0%	0.511
NP0–NP2	83.3%	0.714
NP1–NP2	83.3%	0.657
Window subset		
W0–W1	83.3%	0.683
W0–W2	83.3%	0.740
W1–W2	75.0%	0.765

(b) Downstream success variation

q condition	50-state SR	80-rollout SR
W0 / all	96.0%	96.3%
W0 / NP0	94.0%	93.8%
W0 / NP1	96.0%	95.0%
W0 / NP2	96.0%	93.8%
W1 / all	96.0%	95.0%
W2 / all	92.0%	91.3%

Downstream performance remains comparatively stable. On the 50 official unique states, success varies by 2.0 pp across single-NoisePack conditions and 4.0 pp across window manifests. The paired 80-rollout evaluation gives corresponding ranges of 1.3 and 5.0 pp. W0/all is highest or tied highest in both. Sampling therefore affects the final policy only modestly on this task, with temporal-window sampling contributing more variation than NoisePack choice.

The 50-state evaluation is the primary closed-loop result. The 80-rollout evaluation also varies action-sampling randomness while reusing physical initial states. This mechanism-level audit measures sampling sensitivity on one LIBERO-90 Development Split task under one downstream training seed.

C.4 Single-Task Adaptation Cost

We report a representative per-task cost profile on LIBERO-90 Development Split Task 9 using 8 $\times$ A100-SXM4-80GB GPUs. This systems profile differs from the main evaluation protocol in its Stage 2 schedule.

Profiling configuration. Following Appendix A.2, fit3 uses 24 unique windows with 3 NoisePacks each and selects Top-4 independently per route. Stage 1 runs 108 steps. For profiling, Stage 2 fixes historical composition and runs for 10 epochs (120 steps total) with per-GPU/global batch size 4/32.

Table 25: **Core-stage runtime and peak memory for the single-task profiling configuration.** Stage totals are wall-clock times. Mean step times cover Stage 1 optimization and the Stage 2 critical path. Total runtime also includes setup, data preparation, anchor evaluation, and transitions.

Stage	Wall time (s)	Mean step (s)	Peak memory (GiB)
Stage 0: q computation	185.3	—	71.3 (video route)
Stage 1: functional reuse	339.3	2.758	39.0
Stage 2: target-group refinement	582.2	2.921	33.7

D Historical Adapter Characterization

Before historical reuse, we test whether adapters develop stable and distinguishable internal responses in the shared structured adaptation space. We characterize G_v in the video route through its layerwise and headwise responses. We use these responses as adapter signatures for analysis, while target-loss q remains the candidate-selection criterion.

D.1 Adapter-Specific G_v Signatures

We pass identical common-panel samples through frozen historical adapters H1–H15. FastWAM's 30 layers and 24 heads per layer produce one layer-head G_v pattern per observation and adapter.

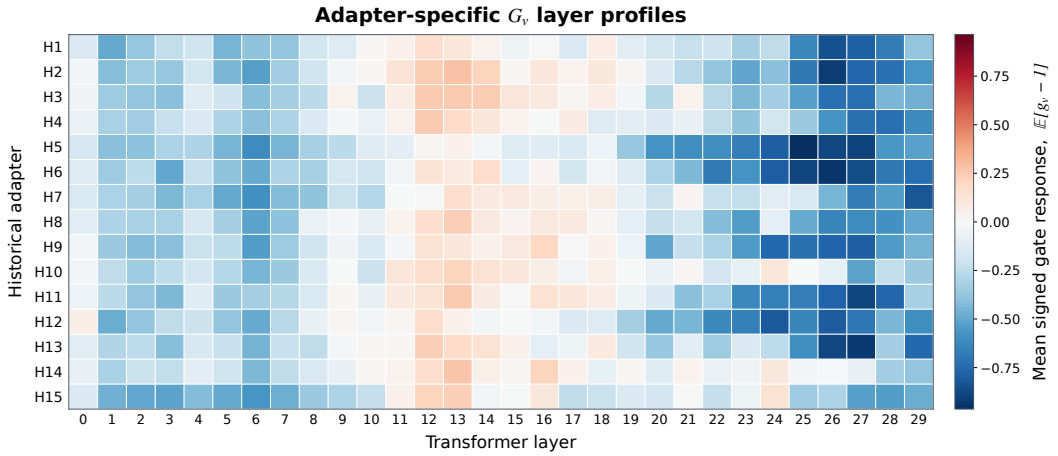


Figure 6: **Layerwise G_v responses of historical adapters.** Rows are H1–H15 and columns are 30 Transformer layers. Color gives the common-panel mean signed response, $\mathbb{E}[g_v - 1]$. Adapters share middle-layer enhancement and later-layer suppression while retaining distinct cross-layer patterns. The plot shows raw responses. Separability experiments remove overall level and scale.

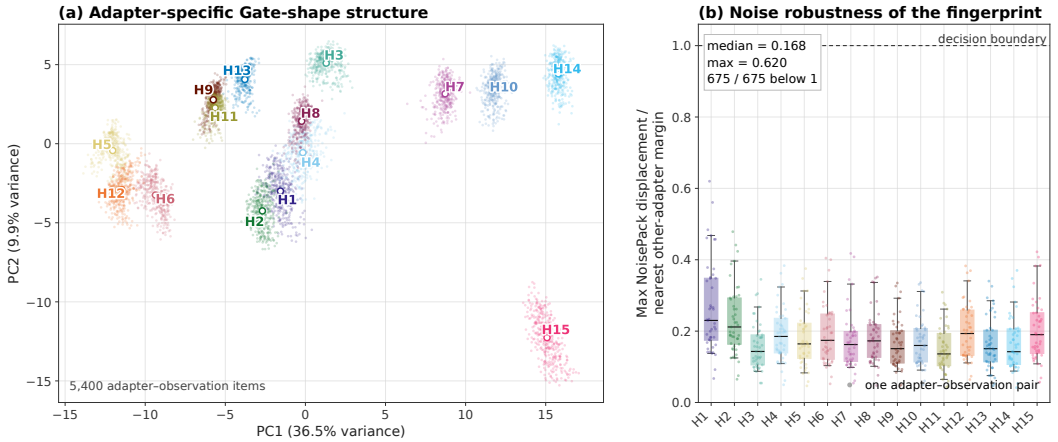


Figure 7: **Separability and noise stability of historical G_v signatures.** (a) PCA projections of layer-shape features from 5,400 adapter–observation instances separate H1–H15. (b) The ratio ρ compares maximum NoisePack displacement with the nearest-other-adapter separation margin. All 675 groups remain below $\rho = 1$. Table 26 and Section D.3 give the quantitative evaluation.

Historical adapters share a broad depth profile while retaining adapter-specific cross-layer structure. Figure 6 shows mostly negative early-layer responses, relative enhancement through the middle layers, and stronger suppression near the final layers. Their locations and magnitudes vary across H1–H15, so a global gate magnitude cannot capture these signatures. We therefore remove overall level and scale from layer-shape features for the cross-task analysis below, while retaining layerwise statistics and full headwise features as complementary views.

D.2 Cross-Task Separability

The separability study uses 15 adapters and 360 common-panel windows, giving 5,400 adapter–observation instances. We first split observations by episode. We then remove one complete observation task from training to test whether signatures remain identifiable across tasks.

ARI (Adjusted Rand Index) and NMI (Normalized Mutual Information) measure agreement between unsupervised clusters and adapter identities, with 1 indicating exact agreement. The Silhouette score increases with within-cluster compactness and between-cluster separation.

Cross-layer structure identifies historical adapters across observation tasks. A global scalar gate magnitude gives Top-1 accuracies of 0.669 and 0.661 under episode-held-out and task-held-out evaluation. Layer-shape features reach 1.000 in both settings and remain at 1.000 when one NoisePack and one task are jointly held out. Without adapter labels, K-means also obtains ARI=NMI=1.000, so its 15 clusters correspond exactly to H1–H15. The permutation test confirms that this separation exceeds random-label results. Historical-adapter differences are therefore encoded primarily in the cross-layer structure of G_v rather than its mean gate magnitude.

D.3 Robustness to Noise Perturbations

We next test signature stability under input noise. Repeating 45 base observations for 15 fixed adapters under 4 independent NoisePacks yields 2,700 samples.

For adapter a and observation o, define

$$\rho_{a,o} = \frac{\max_n \|z_{a,o,n} - \bar{z}_{a,o}\|_2}{\min_{b \neq a} \|\bar{z}_{a,o} - \bar{z}_{b,o}\|_2}, \tag{13}$$

The numerator is the largest noise-induced displacement for one adapter–observation pair, and the denominator is its distance to the nearest other adapter signature. A value $\rho < 1$ means that even the worst perturbation remains within the nearest-other-adapter separation margin.

Noise-induced variation remains smaller than the separation between adapters. We obtain $\mathrm{median}(\rho) = 0.1679$ and $\max(\rho) = 0.6200$, with all 675 adapter–observation groups satisfying $\rho < 1$. Across NoisePacks, same-adapter layer-shape cosine similarity has median 0.9976 and minimum 0.9397. Shape ICC for repeated measurements has median 0.9841. The historical adapters therefore form stable and distinguishable route-level residual signatures that persist across observation tasks and noise, supporting the shared route-wise organization of historical adaptation.

Table 26: **Separability and noise stability of historical G_v signatures.** We compare global scalar magnitude with layer-shape features using classification, clustering, and stability metrics. Task+Noise denotes leave-one-noise-pack-and-task-out evaluation. Top-1 is a fraction. Section D.3 defines the noise settings, and dashes mark unreported metrics.

Evaluation	Scalar	Layer-shape
Episode-held-out Top-1	0.669	1.000
Leave-one-task-out Top-1	0.661	1.000
K-means clustering	—	ARI 1.000, NMI 1.000, Silhouette 0.646
Permutation test	—	Observed 1.000, $p = 0.004975$
Task+Noise held-out Top-1	0.644	1.000
Paired shape cosine	—	Median 0.9976, Min 0.9397
Shape ICC	—	Median 0.9841

E Experience Persistence and Bank Expansion

This appendix supports Section 6. Sections E.1–E.2 distinguish exact single-task folding from group-level shared approximation. Section E.3 analyzes composition consistency across six target groups, and Section E.4 gives the bank expansion protocol after experience accumulates.

E.1 Exact Folding of a Fixed Routed Composition

Parallel residual composition. We first consider one task or any fixed coefficient set. Folding reparameterizes its routed function as a standalone residual adapter independent of the original bank. Section E.2 separately treats compression from multiple task-specific states into one shared entry.

For route $m \in \{V, C, A\}$, consider an injection point with shared input h and historical adapter IDs $S_m = \{i_1, \ldots, i_K\}$. Write the historical and new residuals uniformly as

$$c_j = \bar{\alpha}_{i_j,m}, \quad d_j(\cdot) = d^{\text{hist}}_{i_j,m}(\cdot) \quad (j \le K), \qquad c_{K+1} = \bar{\eta}, \quad d_{K+1}(\cdot) = d^{\text{new}}_m(\cdot; \phi^*_m). \tag{14}$$

The correction to represent is

$$r_m(h) = \sum_{j=1}^{K+1} c_j d_j(h). \tag{15}$$

We fix c_j and require residuals at the same injection point to read a common input and sum in parallel. The input h contains all tensors required at that point and may reflect upstream adaptation; it need not come from an anchor-only forward pass. Applying the construction at every interface yields d^m_{N+1}; the network-equivalence argument below establishes equivalence. For task-wise folding, use $c_j = \alpha^*_{t,i_j,m}$ and $c_{K+1} = \eta^*_t$ instead.

Linear residual B_v. Let $u = \mathrm{LN}(H_v)$ and $d_j(H_v) = W_{\beta,j} u$. Then

$$\sum_j c_j d_j(H_v) = \left(\sum_j c_j W_{\beta,j} \right) u = \bar{W}_\beta \,\mathrm{LN}(H_v), \qquad \bar{W}_\beta = \sum_j c_j W_{\beta,j}. \tag{16}$$

Input and output dimensions are unchanged. With low-rank storage, factors can be concatenated up to the sum of their ranks. Recovering the original rank generally requires approximation. Candidate-specific LayerNorm affine parameters and projection biases should first be absorbed into effective matrices and biases, with the latter combined as $\bar{b}_\beta = \sum_j c_j b_{\beta,j}$.

Bottleneck residuals. Let $u = \mathrm{LN}(H)$. For each candidate, write

$$d_j(H) = U_j \,\mathrm{SiLU}(D_j u), \qquad D_j \in \mathbb{R}^{r_j \times D}, \quad U_j \in \mathbb{R}^{D \times r_j}. \tag{17}$$

Construct

$$\bar{D} = \begin{bmatrix} D_1 \\ D_2 \\ \vdots \\ D_{K+1} \end{bmatrix}, \qquad \bar{U} = \begin{bmatrix} c_1 U_1 & c_2 U_2 & \cdots & c_{K+1} U_{K+1} \end{bmatrix}. \tag{18}$$

Since SiLU acts elementwise,

$$\bar{U} \,\mathrm{SiLU}(\bar{D} u) = \sum_{j=1}^{K+1} c_j U_j \,\mathrm{SiLU}(D_j u) = r_m(H). \tag{19}$$

The folded width is $R = \sum_j r_j$, or $(K+1)r$ for equal widths; lossless representation at the original width is generally impossible. This construction requires the same elementwise activation, no stochastic dropout, and normalization reducible to a common input. Absorb LayerNorm affine parameters and down biases into input projections, and output scales into output projections and their biases. Stack down biases and sum up biases with the corresponding weights. Additional nonlinear or temporal operators that cannot be absorbed must remain as separate parallel residual paths.

Identity-centered gates. Following Section 4.1, G_v uses $(x, V, \ell_j) = (X_v, V_v, \ell_{v,j})$, while G_c uses $(V^b_{\text{ctx}}, V^b_{\text{ctx}}, \ell^b_{c,j})$ for video/action branch $b \in \{v, a\}$. Each ℓ_j includes the full normalization, trunk, and projection. The residual relative to identity is

$$d_j(x, V) = [2\sigma(\ell_j(x)) - 1] \odot V = \tanh\left(\frac{\ell_j(x)}{2} \right) \odot V. \tag{20}$$

Their weighted sum is exactly

$$\begin{aligned} r_m(x, V) &= \left[\sum_j c_j \tanh\left(\frac{\ell_j(x)}{2} \right) \right] \odot V, \\ \bar{g}(x) &= 1 + \sum_j c_j \tanh\left(\frac{\ell_j(x)}{2} \right), \qquad \widetilde{V} = \bar{g}(x) \odot V. \end{aligned} \tag{21}$$

Copying all coefficients and parameters (c_j, ℓ_j) into the new entry yields a multi-generator gate independent of the old bank files. Each residual path preserves its broadcasting, attention mask, and scope. This exact representation generally cannot be compressed back to a single generator of the original structure, and gate values are no longer restricted to $(0, 2)$.

Single-projection approximation. Single-projection compression is an optional approximation. For linear logits $u_j = W_j x + b_j$, if all u_j and $\bar{u} = \sum_j c_j u_j$ are sufficiently small, $\tanh(u/2) = u/2 + O(u^3)$ gives

$$\bar{g}(x) \approx 1 + \frac{1}{2}\sum_j c_j(W_j x + b_j) \approx 2\sigma(\bar{W}x + \bar{b}), \qquad \bar{W} = \sum_j c_j W_j, \quad \bar{b} = \sum_j c_j b_j, \tag{22}$$

Set $b_j = 0$ when there is no bias. The componentwise error is $O(\sum_j |c_j||u_j|^3 + |\bar{u}|^3)$, so both individual and combined logit magnitudes must be checked. This argument does not yield a structurally identical single-projection representation when nonlinear trunks are present. Using the approximation requires validating output error and closed-loop performance.

Proposition (network equivalence for fixed coefficients). For fixed task-specific or shared coefficients, replacing every injection point exactly with the constructions above preserves the full network output when all other computation, module ordering, and external inputs remain unchanged.

Proof. Let $\mathcal{T}_p$ and $\widetilde{\mathcal{T}}_p$, $p = 1, \ldots, P$, be state updates of the original and folded networks in topological order. State $\mathbf{x}_p$ contains all tensors needed subsequently. Base operators are unchanged, and each injection point satisfies a functional identity, so

$$\widetilde{\mathcal{T}}_p(\mathbf{x}) = \mathcal{T}_p(\mathbf{x}). \tag{23}$$

Starting from identical states $\widetilde{\mathbf{x}}_0 = \mathbf{x}_0$, induction gives

$$\widetilde{\mathbf{x}}_p = \widetilde{\mathcal{T}}_p(\widetilde{\mathbf{x}}_{p-1}) = \mathcal{T}_p(\mathbf{x}_{p-1}) = \mathbf{x}_p. \tag{24}$$

All intermediate states and final outputs therefore agree, including the effects of upstream adaptation on later interfaces. □

The result holds in evaluation mode and exact arithmetic. Multi-step sampling also requires identical noise, conditioning, and sampling procedures. Export must preserve masks, caches, and other updated shared parameters, such as input encoders, and floating-point error must be checked. This equivalence applies to a routed composition with fixed coefficients. Restricting folded widths or gate generators may introduce additional structural compression error. Compressing multiple task-specific routed states into shared coefficients is a distinct group-level approximation, discussed in Section E.2.

E.2 From Task-Specific Routed States to a Shared Folding Entry

For a single task, Section E.1 directly provides equivalent folding under its learned coefficients. To immediately persist an entire target group as one standalone bank entry without retraining a full group-level adapter, multiple task-specific routed states must first be converted to shared coefficients:

$$\{\alpha^*_{t,i,m}, \eta^*_t\}_{t=1}^M \longrightarrow \{\bar{\alpha}_{i,m}, \bar{\eta}\}. \tag{25}$$

GSC operates on the shared set S_m. TSC replaces it with $U_m = \bigcup_t S_{t,m}$, aligning by adapter ID and zero-filling unselected coefficients. The following methods construct only the group-shared coefficients; exact folding from Section E.1 is applied afterward.

Simple averaging (F-mean). Take the task mean of effective coefficients and exposure:

$$\bar{\alpha}_{i,m} = \frac{1}{M}\sum_{t=1}^M \alpha^*_{t,i,m}, \qquad \bar{\eta} = \frac{1}{M}\sum_{t=1}^M \eta^*_t. \tag{26}$$

Average effective coefficients α directly, rather than averaging γ and w separately and multiplying. Since $\bar{\alpha}_{i,m} \geq 0$ and $\sum_i \bar{\alpha}_{i,m} = M^{-1}\sum_t \gamma^*_{t,m} \in (0, 2)$, the mean preserves the coefficient

constraints in Section 4.2.2. It equals the mean residual only for a fixed local input, not the mean full-policy output.

Coefficient fitting / tied optimization (F-tied). Freeze the S2 new residual adapter $\boldsymbol{\phi}^*$, anchor, historical bank, and other shared parameters, and fit only cross-task tied coefficients with the parameterization from Section 4.2.2:

$$\bar{\gamma}_m = 2\sigma(\bar{\rho}_m), \qquad \bar{w}_{i,m} = \frac{\exp(\bar{z}_{i,m})}{\sum_{j\in S_m}\exp(\bar{z}_{j,m})}, \qquad \bar{\alpha}_{i,m} = \bar{\gamma}_m \bar{w}_{i,m}. \tag{27}$$

Fix $\bar{\eta}$ to its F-mean value and optimize shared coefficients under the original action loss, optionally initialized by F-mean:

$$\{\bar{\rho}_m^*, \bar{\mathbf{z}}_m^*\}_m = \underset{\{\bar{\rho}_m, \bar{\mathbf{z}}_m\}_m}{\arg\min} \frac{1}{M}\sum_{t=1}^{M} \mathcal{L}_{\mathrm{FM-action}}(\mathcal{D}_t; \bar{\boldsymbol{\alpha}}(\bar{\boldsymbol{\rho}}, \bar{\mathbf{z}}), \boldsymbol{\phi}^*, \bar{\eta})\,. \tag{28}$$

To also fit exposure, introduce $\bar{s}$ and set $\bar{\eta} = \eta_{\mathrm{init}} + (1-\eta_{\mathrm{init}})\sigma(\bar{s})$. After fitting, apply Section E.1 folding, which is exact relative to these shared coefficients.

At the same local input $h_{t,m}$, sharing introduces the residual difference

$$e_{t,m}(h_{t,m}) = \sum_{i\in S_m} (\alpha^*_{t,i,m} - \bar{\alpha}_{i,m}) d^{\mathrm{hist}}_{i,m}(h_{t,m}) + (\eta^*_t - \bar{\eta}) d^{\mathrm{new}}_m(h_{t,m}; \phi^*_m). \tag{29}$$

The error in Section E.2 thus comes from sharing task-specific composition degrees of freedom, not from folding parallel residuals with fixed coefficients in Section E.1. More consistent compositions make a shared folding entry more likely to preserve the current routed experience. If compositions differ substantially, retaining the routed state temporarily and awaiting further group-level experience is more appropriate.

E.3 Composition Consistency Across Target Groups

Figure 8 complements Section 5.3 with task-specific GSC Top-4 coefficients $\alpha = \gamma w$ for three routes and four candidates per route in all six groups.

We compute the mean task-pair distance after aligning each task's effective coefficients to 45 full-bank coordinates, $V/C/A \times H1$–$H15$, with zeros for inactive adapters.

Let $a_t \in \mathbb{R}^{45}$ be the aligned coefficient vector for task t, and M the number of tasks in the group. The L2 distance between two distinct tasks is

$$D_{ts} = \|a_t - a_s\|_2. \tag{30}$$

Averaging over all $\binom{M}{2}$ distinct task pairs yields the main-text metric:

$$\overline{D} = \frac{2}{M(M-1)} \sum_{1\le t<s\le M} D_{ts}. \tag{31}$$

A smaller $\overline{D}$ indicates more similar effective composition coefficients within the group.

E.4 Group-Level Bank Expansion Protocol

Long-term bank expansion follows the function-structured group adaptation in Section 3. Once related tasks and demonstrations exhibit shared operational requirements, we reorganize them as $\mathcal{G}_{N+1}$ and train $H_{N+1} = \{d^V_{N+1}, d^C_{N+1}, d^A_{N+1}\}$ from the common anchor f_{θ_0}. The backbone remains frozen and training uses the historical-adapter action objective. The resulting H_{N+1} is frozen and added to the bank as $\mathcal{B} \leftarrow \mathcal{B} \cup \{H_{N+1}\}$.

Bank expansion thus has two paths at different timescales. For immediate persistence, Sections E.1–E.2 reparameterize the current routed state into a standalone bank entry through folding. Otherwise, related experience can continue to accumulate before training a standard group-level adapter. Since subsequent candidate selection recomputes route-wise transferability from target demonstrations, neither type of new entry requires updating a cross-task neural router.

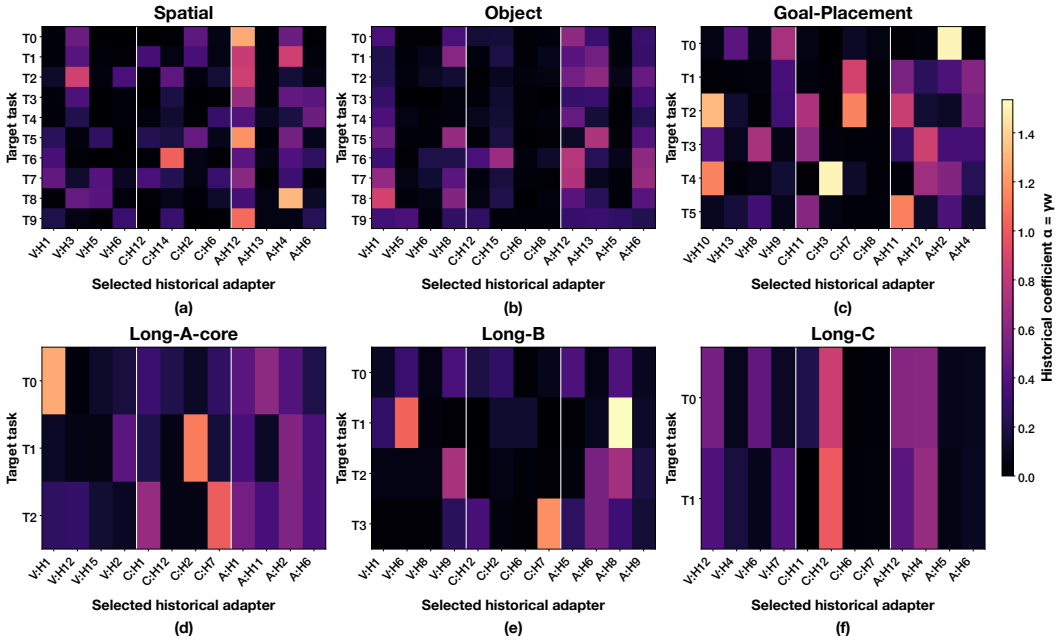


Figure 8: **Task-specific Top-4 historical compositions for all six target groups.** Panels cover Spatial, Object, Goal-Placement, Long-A-core, Long-B, and Long-C. Rows denote tasks, and columns denote selected adapters grouped by video, context, and action routes. Color shows $\alpha = \gamma w$ on a shared scale. Candidate identities vary by group, so columns are not aligned across panels.