ROBOTASKONOMY: DISCOVERING TRANSFER-RELEVANT KINSHIP AMONG ROBOTIC SKILLS

Anonymous authors
Paper under double-blind review

ABSTRACT

Real-world robots are expected to continually acquire new skills to adapt to changing demands and environments. The ability of prior skills to facilitate new skill acquisition, i.e., skill transferability, is critical for improving learning efficiency. However, skill transferability remains poorly understood, and it is unclear which relationships among skills are relevant to transfer and how they should be characterized. In this paper, we present RoboTaskonomy, a framework for characterizing transfer-relevant relationships among robotic skills and discovering how these relationships are structured. We first construct skill relationships from task attributes, fine-grained trajectories, and local motion distributions, then assess their correspondence with measured transfer effects. This analysis identifies a transfer-relevant representation space in which skill relationships align more closely with measured transfer effects. Building on this representation space, we discover a set of behavioral primitives shared across skills. Skills can be represented through combinations of these primitives, with stable structure across task contexts and unseen skills emerging as novel compositions. We further leverage this structure to guide prior-skill selection in continual learning, significantly improving new-skill acquisition in both simulation and real-world tasks.

1 INTRODUCTION

A real-world robot is expected to continually acquire new skills and adapt to evolving operational demands across diverse environments (Wan et al., 2024; Liu et al., 2024; Wołczyk et al., 2021). When new skills need to be acquired, previously learned skills can serve as effective stepping stones, allowing the robot to leverage existing capabilities and achieve superior performance (Yijie Guo, 2025). However, the extent to which a previously learned skill facilitates the acquisition of a new skill (i.e., skill transferability) varies with the skills involved. Understanding this variation requires identifying the similarities and differences between skills that matter for transfer. Yet how these transfer-relevant relationships should be represented remains unclear.

Crucially, transfer-relevant relationships are not known a priori and depend on how robotic skills are characterized. Such characterizations span multiple levels of abstraction, from coarse task semantics to fine-grained motion trajectories, each preserving different aspects of behavior. The key is therefore to identify what information should be retained or abstracted away for the resulting relationships to reflect empirical transferability. Beyond identifying a transfer-relevant representation, a further question is whether the relationships it captures can be understood through recurring behavioral components shared across skills. Uncovering this structure could help explain how previously acquired skills collectively support the acquisition of new ones.

In this work, we present RoboTaskonomy, a framework to represent and organize transfer-relevant relationships across robotic skills. To begin with, we quantify transfer effects by measuring success-rate gains in cross-skill adaptation (Fig. 1(a)). Using these measurements as an empirical reference, we analyze how skill relationships derived from different representations correspond to measured transfer effects. As illustrated in Fig. 1(b), we construct skill relationships across three levels of abstraction: coarse task attributes, fine-grained trajectories, and local motion distributions from complementary trajectory features. We find that relationships derived from local motion distributions align more closely with measured transfer effects, supporting an abstraction that preserves recurring execution characteristics while reducing trajectory-specific detail.

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.

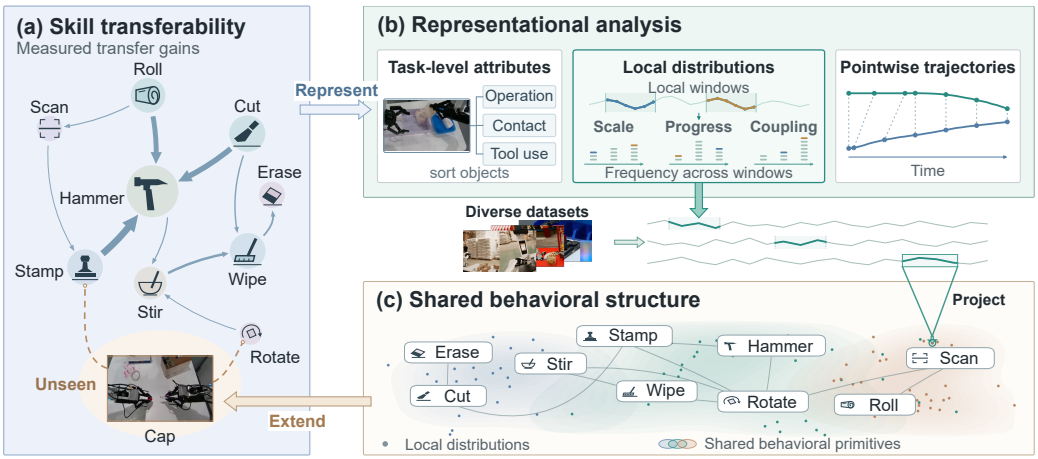


Figure 1: Overview of RoboTaskonomy. (a) Cross-skill adaptation measures skill transferability. (b) Comparison with measured transfer identifies local motion distributions as a transfer-relevant representation. (c) Factorizing these distributions reveals shared behavioral primitives whose stable compositions define *Skill Kinship*, while unseen skills emerge as novel combinations.

Building on this transfer-relevant representation, we further advance our understanding of skill transferability by uncovering the shared behavioral structure underlying skill relationships. To extract these components, we apply nonnegative factorization to local motion distributions, terming them behavioral primitives. As shown in Fig. 1(c), these primitives define a common behavioral basis in which each skill is represented by a compact composition that remains stable across task contexts and retains strong correspondence with measured transfer effects. Analogous to biological kinship through shared genetic traits, we refer to this compositional structure as *Skill Kinship*.

This shared basis naturally extends to unseen skills, which can be expressed through the same behavioral components and related to existing skills through their compositions, as depicted in Fig. 1(a). Beyond organizing skill relationships, *Skill Kinship* also provides an inductive bias for continual learning. We use it to guide the initialization of new skills with prior skills expected to provide more effective transfer. This relation-guided initialization consistently improves final performance across simulated and real-world manipulation tasks, demonstrating the practical value of transfer-relevant skill structure for continual skill acquisition.

2 Related Works

Task relationships and transferability. Characterizing transferability calls for relationships that capture how learning one task affects another. Taskonomy maps measured transfer among visual tasks (Zamir et al., 2018), while Task2Box models asymmetric relationships on the same benchmark (Daroya et al., 2024). Multi-task learning studies identify tasks that benefit from shared training (Standley et al., 2020; Fifty et al., 2020). Task2Vec and representation-similarity methods use task embeddings and model responses to select pretrained models (Achille et al., 2019; Dwivedi & Roig, 2019). For robotic tasks, SRSA estimates transfer among assembly skills (Yijie Guo, 2025). In this work, we focus on identifying what behavioral information should be retained to characterize transfer-relevant relationships among robotic skills.

Relationships across robotic skills. Prior work has characterized relationships among robotic skills at different levels of abstraction, ranging from task-level attributes to detailed execution trajectories. Robotic skill taxonomies describe manipulation through task-level motion, coordination, and process attributes (Paulius et al., 2020; Krebs & Asfour, 2022; Johannsmeier et al., 2025). At a finer granularity, trajectory-based methods use dynamic time warping (DTW) to align demonstrations while retaining their temporal motion structure (Sakoe & Chiba, 1978; Vakanski et al., 2012). We compare these representations against measured transfer effects and introduce local motion distributions as an intermediate abstraction to identify transfer-relevant structures.

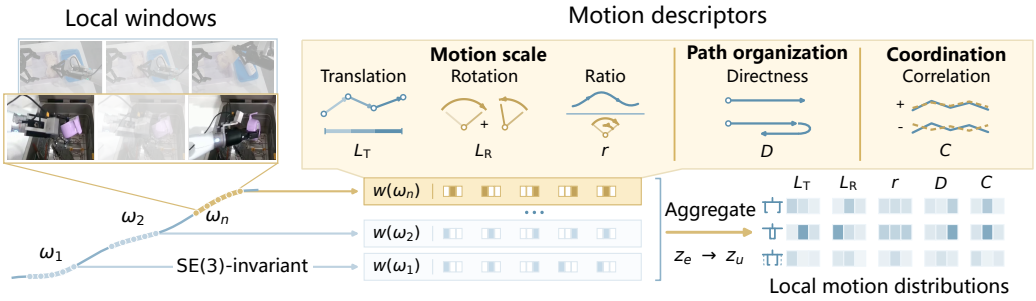


Figure 2: Local motion distributions. Each fixed-duration window is encoded by five discretized motion descriptors and a gripper context. Window records are aggregated into normalized distributions z_e per demonstration, then averaged equally within skill u to obtain its local motion distribution z_u.

Continual robot learning. Continual robot learning involves retaining acquired skills and transferring useful experience to new tasks (Wang et al., 2026; Meng et al., 2025). Continual World and LIBERO evaluate forward transfer in sequential robotic control and manipulation settings (Wołczyk et al., 2021; Liu et al., 2024). Regularization and replay provide general mechanisms for retaining earlier knowledge (Kirkpatrick et al., 2017; Chaudhry et al., 2019). CLARE uses internal feature similarity to guide adapter reuse and expansion, with reconstruction-based routing at inference (Römer et al., 2026). We use behavioral composition similarity to select prior skills for initialization, explicitly grounding continual knowledge reuse in transfer-relevant skill relationships.

3 Representing Transfer-Relevant Relationships

Robotic skills are temporally extended interaction processes that couple high-level task intent with low-level execution realization. Representations at either extreme may therefore be insufficient for characterizing transfer: task-level descriptions can obscure reusable execution structure, whereas raw trajectories retain realization-specific variations. We further introduce local motion distributions between them to preserve recurring execution characteristics while abstracting away trajectory-specific details. The resulting skill relationships are then compared with empirical transfer gains to identify which level of abstraction most closely reflects measured transferability.

3.1 Skill Representations across Abstraction Levels

Task-level descriptions. A coarse comparison describes each skill using its named operation and overall behavioral properties, without comparing the specific motions that occur during execution. For operation semantics, we encode canonical operation phrases using frozen Word2Vec (Mikolov et al., 2013) and compute cosine similarity between normalized embeddings. These phrases capture what operation is performed rather than how it unfolds. To characterize the latter at the task level, we manually adapt the motion taxonomy of Paulius et al. (2020) to each task's principal operation and functional arm, and compare the resulting binary codes using normalized Hamming similarity.

Pointwise trajectories. At the finest level, we represent each execution as a time-ordered sequence of end-effector positions and orientations, describing motion at individual time steps. To construct skill relationships, we align demonstration trajectories using dynamic time warping (DTW) and compare their corresponding poses. The negative mean distance across demonstration pairs defines the skill-relation score. Temporal alignment accommodates differences in execution speed while retaining differences in position and orientation.

Local motion distributions. To construct an intermediate representation along this abstraction spectrum, we characterize each demonstration through the distribution of its local motion characteristics. We first extract fixed-duration sliding windows ω from demonstration trajectories and retain active segments. Each window spans multiple motion increments and is characterized along three complementary aspects of execution: motion scale, path organization, and kinematic coordination. As illustrated in Fig. 2, these aspects are quantified by five measurements: translational

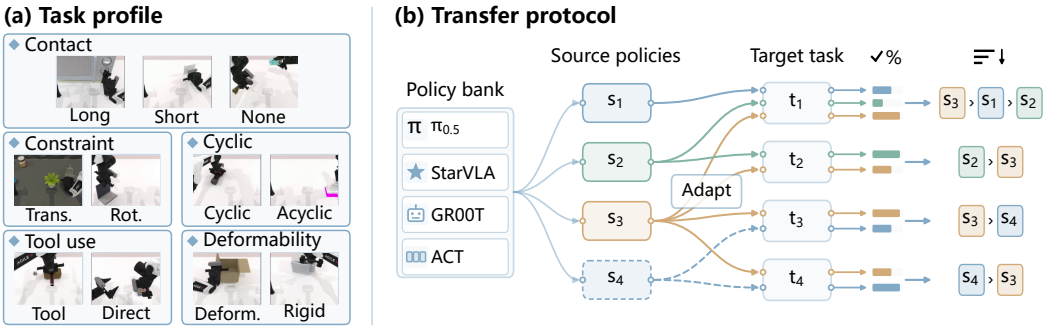


Figure 3: Task profile and transfer protocol. (a) Complementary dimensions in task design. (b) Source policies share an initialization within each family and are separately adapted to targets.

and rotational motion scales, their relative ratio, motion directness, and cross-component correlation. Together, these measurements provide a compact characterization of local motion that reduces sensitivity to pointwise fluctuations while preserving complementary kinematic properties. For end-effector position $\mathbf{p}_i$ and orientation R_i within a window, we first compute the translational and rotational increment magnitudes and aggregate them into the motion descriptor $\phi(\omega)$:

$$u_i = \|\mathbf{p}_{i+1} - \mathbf{p}_i\|_2, \qquad v_i = \left\|\mathrm{Log}(R_i^\top R_{i+1})^\vee\right\|_2, \tag{1}$$

$$\phi(\omega) = \left[\sum_i u_i,\ \sum_i v_i,\ \frac{\sum_i u_i}{\sum_i v_i},\ D(\omega),\ \mathrm{Corr}(\{u_i\},\{v_i\})\right]. \tag{2}$$

Motion directness D characterizes local path organization in terms of translational and rotational efficiency, spatial linearity, and motion reversals. The final term measures the extent to which translational and rotational motions co-vary over the same interval, capturing their kinematic coordination. The resulting descriptor $\phi(\omega)$ is invariant to global $\mathrm{SE}(3)$ transformations of the pose sequence, characterizing local execution independently of workspace placement and orientation.

As for similar local motions can occur under different interaction contexts, we additionally associate each window with a gripper context $g(\omega) \in \mathcal{G}$, corresponding to predominantly open, predominantly closed, or transitional states. Each motion measurement is discretized into three ordered bins using a feature-specific quantizer Q_a, fitted on external motion data and then frozen. Let $\mathcal{B} = \{b_1, b_2, b_3\}$ denote these bins. Each window is represented as:

$$w(\omega) = \Big(g(\omega), \big[Q_a\big(\phi_a(\omega)\big)\big]_{a\in\mathcal{A}}\Big) \in \mathcal{G} \times \mathcal{B}^{|\mathcal{A}|}. \tag{3}$$

This conditioning provides execution context without requiring semantic task-phase annotations. We then aggregate these window-level representations into a distributional description, discarding exact temporal order while retaining the prevalence of different local motion characteristics. For a demonstration e, normalized joint histograms of gripper context and each discretized motion feature are concatenated with equal feature weighting to form the local motion distribution z_e. Averaging z_e across valid demonstrations yields the skill-level distribution z_u, providing a common statistical representation for comparing local execution across skills. Details are provided in Appendix A.

3.2 Measurements on Transfer Effects

Task profile. To systematically measure transfer effects across diverse skills, we design a task suite with broad behavioral variation and controlled semantic overlap. As shown in Fig. 3(a), the tasks vary along five behavioral dimensions: contact regime (sustained, brief, or absent), kinematic constraint (translation- or rotation-dominated), motion cyclicity (repetitive or one-off), tool mediation (tool-assisted or direct), and object mechanics (rigid objects or folding surrogates). All simulation tasks are built on RoboTwin (Chen et al., 2025), which we extend with additional assets and task variants. Our main study with $\pi_{0.5}$ (Black et al., 2025) evaluates six source skills for each of 24 target skills, yielding 144 directed transfer pairs. Additionally, We evaluate ACT (Zhao et al., 2023), GR00T-N1.7-3B (Bjorck et al., 2025), and StarVLA (Community, 2026) to assess whether the observed transfer patterns generalize across policy families.

Table 1: Transfer gains and target-only success (Base, %) from the evaluated source-target pairs.

Target	Base (%)	Evaluated sources and gains (pp), in decreasing gain order					
Adjust bottle	90	**Roller** +10	**Bottl** +10	**ShakeH** +10	Lift +8	Dual +8	Knob −86
Hammer block	0	**Stamp** +86	Cut +74	Roll +68	Rank +64	Staple +50	Push +38
Close microwave	92	**Rotate** +4	Lift 0	Open −6	Fold −10	Hang −12	Knob −80
Cut bread	94	**Hammer** +6	**Push** +6	**Roll** +6	**Stamp** +6	**Wipe** +6	Staple +4
Drag waterer	92	**Roll** −4	**Wipe** −4	Card −8	CanPot −10	Sweep −12	Push −14
Erase board	48	**Wipe** +30	Drag +20	Sweep +12	Hammer +2	Roll +2	Push 0
Fold box	12	**Hang** +14	**Bowls** +14	Close +10	Stk3 +4	Stk2 0	Open −8
Handover block	12	**Dual** +32	Hang +24	Stk2 +18	Fold +10	Basket +6	CanPot −6
Hang mug	14	**Fan** +16	Rotate +12	Left +4	Fold −10	Open −12	Close −14
Lift pot	72	**CanPot** +20	Dual +18	Close +16	Hang +16	Laptop +16	Bottl +10
Move pill bottle	62	**CanPot** +16	Cans −6	Card −14	Roll −24	Right −26	Push −46
Open microwave	0	**Close** +32	Drag +14	Laptop +10	Rotate +10	Fold +4	Hang +2
		...					

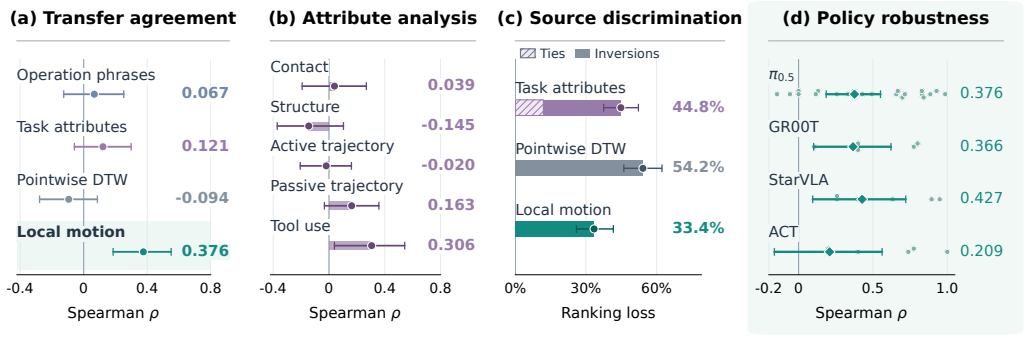


Figure 4: Transfer ranking analysis. (a) Correlations between different skill representations and observed target success. (b) Correlations for individual task attributes. (c) Source discrimination via tie and inversion losses. (d) Local-motion ranking agreement across policies.

Transfer protocol. As shown in Fig. 3(b), all source policies within a policy family share the same initialization and are trained separately on their source-skill demonstrations. Each trained source policy then initializes a restricted adaptation to the target, where most parameters are frozen, and the resulting target success rate measures its transfer effect. We then examine how well different representations of skill relationships reflect the measured transfer outcomes. For each target, we rank the candidate sources in two ways: by their representation-derived scores and by their observed target success rates. Let $\mathbf{r}_t$ and $\mathbf{T}_t$ denote these two sets of scores, respectively. We compute their Spearman correlation for each target and average the correlations across targets for which the correlation is defined:

$$\overline{\rho} = \frac{1}{|\mathcal{T}_{\text{valid}}|} \sum_{t \in \mathcal{T}_{\text{valid}}} \text{Spearman}(\mathbf{r}_t, \mathbf{T}_t), \tag{4}$$

where $\mathcal{T}_{\text{valid}}$ denotes the set of targets with a defined Spearman correlation. All representation methods are evaluated on the same source–target pairs. Additional details on task construction, training settings, and statistical treatment are provided in Appendix B.1.

3.3 Transfer Relevance of Skill Representations

Transfer gains. As shown in Table 1, the evaluation reveals asymmetric, source-dependent transfer. Closing the microwave helps opening, whereas the reverse transfer hurts performance. For a fixed target, sources can also provide different gains: board erasing benefits more from table wiping than from sweeping or pushing blocks. These observations suggest that behavioral commonalities may identify useful sources, motivating the feature-based ranking comparison below.

Agreement with transfer. As shown in Fig. 4(a), local motion distributions achieve the strongest correspondence with source rankings ($\overline{\rho} = 0.376$), while the correlation intervals of the other representations include zero. This contrast reveals a clear effect of representation granularity. Task-level descriptions are too coarse to capture within-skill execution variation, with some attributes unable

to distinguish candidate sources at all and thus producing undefined correlations in Fig. 4(b). At the other extreme, pointwise DTW preserves detailed geometric differences between individual executions, but these realization-specific differences align only weakly with source usefulness. Local motion distributions occupy an intermediate level of abstraction: they suppress exact poses and long-range trajectory arrangement while preserving recurring motion characteristics and their prevalence across executions. These results suggest that transfer-relevant skill relationships are better exposed by abstracting away trajectory-specific details without discarding local execution structure. Ablations on components of local motion distributions are provided in Appendix B.2.

Source discrimination and robustness. We further examine how different representations distinguish candidate sources with unequal observed transfer outcomes, as depicted in Fig. 4(c). Task-level attributes leave a subset of source pairs indistinguishable, whereas pointwise DTW more often assigns an ordering inconsistent with their observed transfer performance. Local motion distributions distinguish all evaluated pairs while producing the fewest pairwise ordering disagreements. As shown in Fig. 4(d), similar positive correspondence is observed across pretrained VLA policies, suggesting that the relation between local motion distributions and transfer rankings is consistent across different pretrained policy families. The policy trained from scratch (ACT) shows weaker and more variable correspondence across targets, likely due to the absence of stable representations from large-scale pretraining. Details on ranking-loss computation and success-rate gains from source selection are provided in Appendix B.3.

4 Uncovering Shared Behavioral Structure

Within the transfer-relevant space spanned by local motion distributions, each skill is characterized by the prevalence of local execution patterns defined over a common set of motion descriptors. As these patterns recur across skills with different prevalence, they can give rise to coordinated variation across multiple distribution dimensions. This suggests that the high-dimensional distribution space may be organized by a smaller set of shared behavioral factors, with individual skills represented by different compositions of these factors. This view is consistent with evidence for shared coordination components in motor behavior (d'Avella & Bizzi, 2005).

We therefore hypothesize that each skill distribution can be approximated by a nonnegative combination of shared components, $z_u \approx m_u H$, where H defines a common behavioral basis and m_u specifies the skill-specific composition. Under this formulation, the diversity of skill execution can be characterized through different combinations of a shared behavioral repertoire, while ***Skill Kinship*** is expressed by component sharing and relative contributions. We examine this structure through three properties: component recurrence across skill subsets, composition reproducibility for seen and held-out skills, and the transfer relevance of composition-induced relationships.

4.1 Discovering Shared Behavioral Primitives

To instantiate this compositional hypothesis, we seek a shared basis whose components capture recurring parts of local execution and jointly reconstruct skill distributions. We refer to these components as behavioral primitives: each primitive describes a recurring motion pattern that may contribute to multiple skills. This motivates a nonnegative additive model, in which multiple primitives can jointly compose a skill and the same primitive can be reused across different skills. Stacking N distributions z_u of dimension d into $X \in \mathbb{R}_+^{N\times d}$, we discover K shared behavioral primitives via KL-NMF factorization (Lee & Seung, 1999):

$$(\widehat{W}, \widehat{H}) \approx \underset{W,H\geq 0}{\arg\min}\, D_{\mathrm{KL}}(X\|WH), \qquad H_{k,:} = \widehat{H}_{k,:}/\|\widehat{H}_{k,:}\|_1. \tag{5}$$

Each unit-mass row of $H \in \mathbb{R}_+^{K\times d}$ specifies a primitive's behavioral content; its coefficient specifies the contribution to a skill distribution. The primitives describe execution statistics rather than discrete executable actions. We fit $K = 6$ components from local samples of 50K AgiBot episodes (Bu et al., 2025), with fixed quantizers and no transfer outcomes used in fitting or model selection.

The six primitives discovered by KL-NMF, denoted M0–M5, exhibit physically interpretable patterns of motion scale, path organization, and translation–rotation coordination. As shown in Fig. 5(a), M0 shows characteristics of repositioning: directed motion with moderate translation

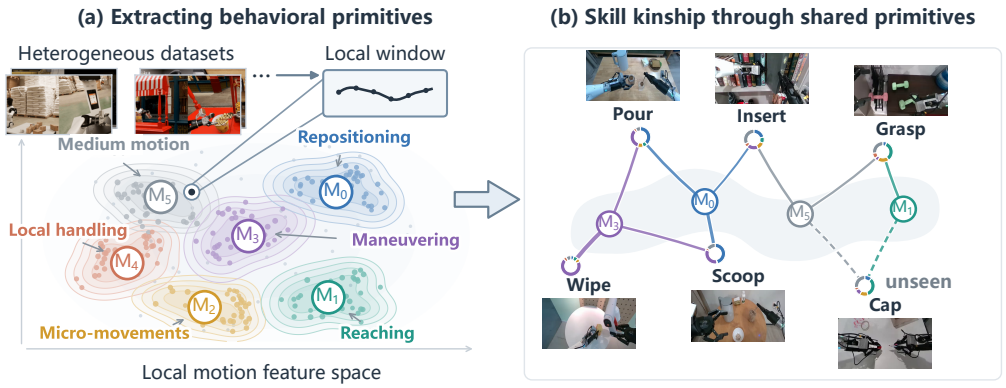


Figure 5: Skill kinship through shared primitives. (a) Discover shared behavioral primitives from local motion distributions. (b) Skills are related through overlapping primitive compositions; dashed links show an unseen skill represented in the same basis.

and prominent rotation. M1 and M3 resemble reaching and maneuvering, respectively, with large, directed translation and rotation and strong coupling. M2 and M4 exhibit small, less directed movements characteristic of micro-movements and local handling, respectively, while M5 captures coordinated translation and rotation at intermediate scales. Across 30 task-bootstrap refits, the recovered components remain highly consistent with the original basis (with median matched cosine similarity over 0.95), indicating stable recurrence across task mixtures. Together, these patterns suggest that, within the transfer-relevant behavioral space, diverse skills share a common repertoire, differing mainly in how its components are combined and weighted.

4.2 Revealing Skill Kinship through Primitive Composition

The shared repertoire identified above allows us to relate skills through how they combine and weight the same behavioral primitives. Representing each skill u by its normalized primitive composition m_u, we quantify pairwise *Skill Kinship* by the cosine similarity between compositions:

$$R_{\text{basis}}(s,t) = \frac{m_s^\top m_t}{\|m_s\|_2 \|m_t\|_2} = \sum_{k=1}^{K} \frac{m_{s,k} m_{t,k}}{\|m_s\|_2 \|m_t\|_2}. \tag{6}$$

Each term quantifies a shared primitive's contribution to the relation, weighted by its relative contribution to both skills. We obtain m_u by projecting local-motion distributions onto the fixed basis H, normalizing the resulting coefficients, and averaging them across demonstrations:

$$w_e \approx \arg\min_{w \geq 0} D_{\text{KL}}(z_e \| wH), \qquad \widetilde{w}_e = w_e / \|w_e\|_1, \qquad m_u = |\mathcal{E}_u|^{-1} \sum_{e \in \mathcal{E}_u} \widetilde{w}_e. \tag{7}$$

Here, $\mathcal{E}_u$ is the demonstration set and $m_{u,k} \geq 0$, with $\sum_k m_{u,k} = 1$. As shown in Fig. 5(b), Pour and Scoop share M0 and M3, whereas Wipe places greater weight on M3. Insert also shares M0 with Pour and Scoop but combines it with M5. These examples illustrate the overlapping structure of *Skill Kinship*: a skill can relate to different skills through different shared components, while its full composition preserves their behavioral distinctions.

For the shared basis to generalize beyond the skills used for fitting, held-out skills should admit reproducible compositions under the same basis. We evaluate this property on five fitted and seven held-out tasks. Using disjoint subsets of 25 demonstrations, both groups yield nearly identical compositions and consistent kinship rankings, with median task-level JS divergence below 0.002 bits and median ranking agreement above 0.97. As an example, the held-out *Cap* skill, excluded from dictionary and quantizer fitting, exhibits a similar composition over M1 and M5 to the *Grasp* skill (Fig. 5(b)). These results show that unseen skills can be reproducibly situated within the shared behavioral repertoire through both their compositions and kinship with other skills.

Finally, we examine whether the compositional relations retain the transfer-relevant structure identified in the original representation space. When evaluated against the measured transfer

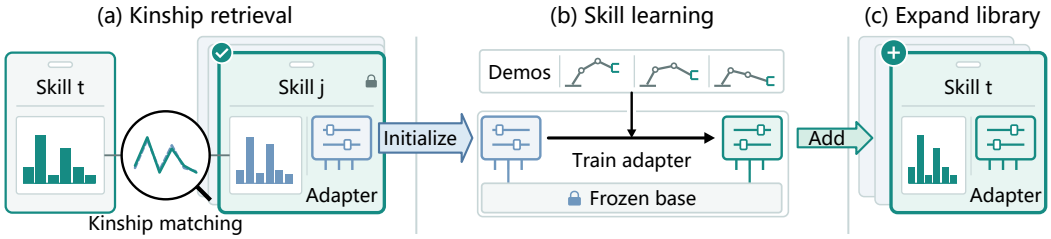


Figure 6: Overview of relation-guided learning (ReL). (a) Retrieve the closest prior skill by composition similarity. (b) Inherit its adapter parameters and train an independent copy on the new task. (c) Expand the library with the new skill's composition, trained adapter, and router.

outcomes, combining composition-based kinship with local-motion similarity further improves transfer-ranking correlation to $\overline{\rho}_\delta = 0.480$. Overall, these findings support a shared compositional organization of robotic skills that is stable across skills and preserves transfer-relevant relationships. Additional validation results and experimental details are provided in Appendix C.

5 Relation-Guided Skill Acquisition

Having identified transfer-relevant skill relationships and uncovered their shared compositional structure, we use this structure to guide continual skill acquisition. Specifically, *Skill Kinship* allows us to select a closely related prior skill as a starting point for learning. This leads to *relation-guided learning* (ReL), where the selected skill's adapter parameters initialize an independent branch that is subsequently trained on the current demonstrations.

5.1 Relation-Guided Initialization

Consider a continual sequence of manipulation skills $\mathcal{T} = (T_1, \ldots, T_N)$, where $\mathcal{D}_t$ contains the demonstrations available for learning skill T_t. For each incoming skill, we obtain its composition m_t by projecting its demonstrations onto the fixed behavioral basis H and averaging the normalized projections, following Eq. 7. The quantizers and H remain frozen throughout the sequence, ensuring that all skills are represented in the same compositional space. For $t > 1$, we use the compositional relation $R_{\text{basis}}(s,t)$ in Eq. 6 to select a previously learned skill for initialization. Specifically, we choose the prior skill with the highest cosine similarity in composition and initialize a new adapter branch from its learned parameters. Let $\Phi_s^\star$ denote the adapter parameters learned for skill T_s. The source selection and corresponding adapter initialization are defined as follows:

$$d_t \in \arg\max_{s<t} R_{\text{basis}}(s,t), \qquad \Phi_t^{(0)} \leftarrow \text{copy}(\Phi_{d_t}^\star). \tag{8}$$

Here d_t indexes the selected source. The behavioral composition thus serves as a retrieval key for selecting which previously learned policy parameters to reuse; neither transfer outcomes nor future-task information enters this decision. For $t = 1$, where no prior source is available, we initialize a fresh adapter. In all cases, the resulting task-specific branch is then optimized only on $\mathcal{D}_t$:

$$\Phi_t^\star \approx \arg\min_{\Phi} \mathbb{E}_{(\mathbf{o},\mathbf{a})\sim\mathcal{D}_t} \left[\mathcal{L}_{\text{FM}}\left(\theta_0, \Phi; \mathbf{o}, \mathbf{a}\right)\right], \tag{9}$$

where $\mathcal{L}_{\text{FM}}$ denotes the flow-matching loss, with $\mathbf{o}$ and $\mathbf{a}$ representing the observations and corresponding actions. The base policy θ_0 and all previously learned adapters remain frozen during optimization. After training, we fit a task-specific reconstruction autoencoder $\Psi_t^\star$ to the policy features of task T_t for layerwise inference-time routing, where each layer activates the adapter with the lowest raw reconstruction error. The resulting $(m_t, \Phi_t^\star, \Psi_t^\star)$ is then registered in the skill library, with m_t used to retrieve source skills for future tasks and $\Psi_t^\star$ used for inference-time routing. Implementation details are provided in Appendix D.

5.2 Continual Learning Experiments

Experimental setup. We evaluate continual skill acquisition in both simulation and real-world settings using $\pi_{0.5}$ (Black et al., 2025) as the base policy. As shown in Fig. 7, the simulation

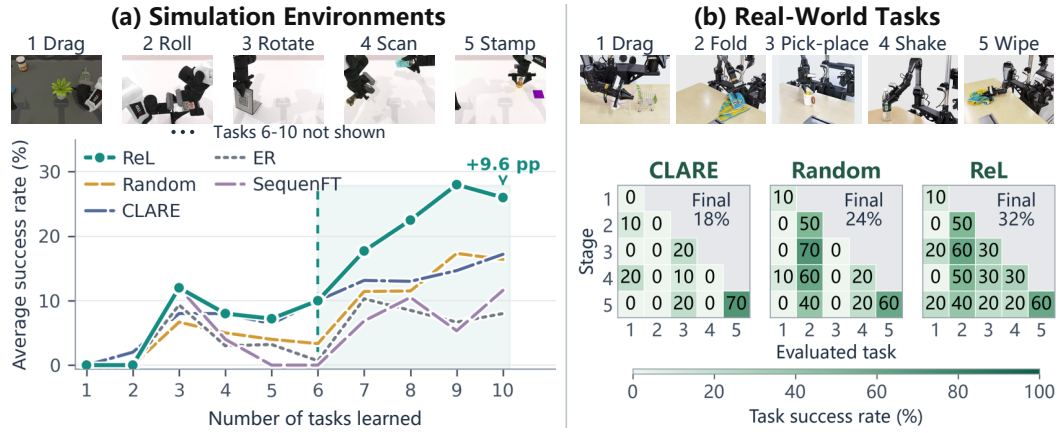


Figure 7: Continual learning in (a) simulation and (b) real-robot experiments. Success rate is averaged over learned tasks on simulators and reported by task and stage on real-world tasks.

benchmark consists of a ten-task stream covering object translation and reorientation, bimanual coordination, and tool use. The real-world benchmark contains five tasks on an AgileX dual-arm robot: cart dragging, towel folding, object placement, bottle shaking, and table wiping.

We compare ReL against baselines representing different knowledge-reuse strategies. A random source-selection baseline (i.e., Random) uses the same module-inheritance framework as ReL, isolating the benefit of relation-guided source selection. CLARE (Römer et al., 2026) selectively reuses or expands task-specific modules based on feature statistics, while sequential fine-tuning (i.e., SequenFT) and experience replay (i.e., ER) represent shared-parameter continual learning with and without data replay (Chaudhry et al., 2019). After learning each new task, we evaluate all tasks encountered so far, measuring both acquisition of the current skill and performance over the accumulated task set. Additional metrics and evaluation details are provided in Appendix E.

Simulation results. ReL improves final success over Random and CLARE by 9.6 and 8.8 percentage points, respectively, while also outperforming the two shared-parameter baselines. Compared with Random, ReL achieves both stronger new-task acquisition and higher mean success throughout the learning stream. As shown in Fig. 7(a), its performance remains comparable to the strongest baselines in the early stages and develops a clearer advantage as more skills are accumulated. This trend is consistent with relation-guided inheritance: as the skill library expands, ReL can select among a broader set of prior skills using their behavioral relations, providing more suitable initialization for incoming tasks. The consistent improvement over the matched Random baseline highlights the benefit of relation-guided source selection.

Real-robot results. As shown in Fig. 7(b), ReL achieves comparable or higher success than Random and CLARE on most tasks, both when each task is first acquired and at the end of the continual stream. Although the better-performing baseline differs across tasks, ReL maintains consistently strong performance and attains the highest final success overall. Together with the simulation results, these findings support the practical value of *Skill Kinship* in capturing transfer-relevant skill relationships and guiding knowledge reuse during continual skill acquisition.

6 Conclusion and Discussion

We presented RoboTaskonomy, a framework for uncovering transfer-relevant skill relationships through behavioral abstraction. By connecting measured transfer effects to behavioral representations, we identify local motion distributions as a transfer-relevant representation and further reveal a shared behavioral basis in which skills exhibit stable compositional structure. These relationships provide a useful prior for knowledge reuse, improving continual skill acquisition under different settings. Future work could explore skill transferability across different robotic embodiments. Understanding such relationships could further improve the learning efficiency and generalization of robotic policies in cross-embodiment settings.

AI USE STATEMENT

In this work, we have not used generative AI tools for generating synthetic data sets, helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods, supporting qualitative and thematic data analysis, and interpreting results, and assisting with translation, cleaning and reformatting dataset are not applicable to this work. Additionally, we used generative AI tools for editing a research paper to improve readability. We have reviewed all AI-assisted work. We checked LLM-generated sentences and take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To support reproducibility, we provide the feature definitions and distribution construction procedures in Appendix A, and the transfer protocols, adaptation hyperparameters, and evaluation panels in Appendix B. Dictionary fitting, demonstration projection, and composition validation are detailed in Appendix C. The continual-learning implementation is described in Appendix D, with task sequences, evaluation metrics, and simulation and real-robot evaluation protocols in Appendix E.

REFERENCES

Alessandro Achille, Michael Lam, Rahul Tewari, Avinash Ravichandran, Subhransu Maji, Charless C Fowlkes, Stefano Soatto, and Pietro Perona. Task2vec: Task embedding for meta-learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6430–6439, 2019.

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

Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y Galliker, et al. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In *Conference on Robot Learning*, 2025.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xuan Hu, Xu Huang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025.

Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486*, 2019.

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

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

Rangel Daroya, Aaron Sun, and Subhransu Maji. Task2box: Box embeddings for modeling asymmetric task relationships. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28827–28837. IEEE, 2024.

Andrea d'Avella and Emilio Bizzi. Shared and specific muscle synergies in natural motor behaviors. *Proceedings of the national academy of sciences*, 102(8):3076–3081, 2005.

Kshitij Dwivedi and Gemma Roig. Representation similarity analysis for efficient task taxonomy transfer learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Christopher Fifty, Ehsan Amid, Zhe Zhao, Tianhe Yu, Rohan Anil, and Chelsea Finn. Measuring and harnessing transference in multi-task learning. *arXiv preprint arXiv:2010.15413*, 2020.

Lars Johannsmeier, Samuel Schneider, Yanan Li, Etienne Burdet, and Sami Haddadin. A process-centric manipulation taxonomy for the organization, classification and synthesis of tactile robot skills. *Nature Machine Intelligence*, 7(6):916–927, 2025.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

Franziska Krebs and Tamim Asfour. A bimanual manipulation taxonomy. *IEEE Robotics and Automation Letters*, 7(4):11031–11038, 2022. doi: 10.1109/LRA.2022.3196158.

Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *nature*, 401(6755):788–791, 1999.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

Yuan Meng, Zhenshan Bing, Xiangtong Yao, Kejia Chen, Kai Huang, Yang Gao, Fuchun Sun, and Alois Knoll. Preserving and combining knowledge in robotic lifelong reinforcement learning. *Nature Machine Intelligence*, 7(2):256–269, 2025.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

David Paulius, Nicholas Eales, and Yu Sun. A motion taxonomy for manipulation embedding. In *Robotics: Science and Systems*, 2020.

Ralf Römer, Yi Zhang, Yuming Li, and Angela P Schoellig. Clare: Continual learning for vision-language-action models via autonomous adapter routing and expansion. *IEEE Robotics and Automation Letters*, 2026.

Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

Trevor Standley, Amir Zamir, Dawn Chen, Leonidas Guibas, Jitendra Malik, and Silvio Savarese. Which tasks should be learned together in multi-task learning? In *International conference on machine learning*, pp. 9120–9132. PMLR, 2020.

Aleksandar Vakanski, Iraj Mantegh, Andrew Irish, and Farrokh Janabi-Sharifi. Trajectory learning for robot programming by demonstration using hidden markov model and dynamic time warping. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1039–1052, 2012.

Weikang Wan, Yifeng Zhu, Rutav Shah, and Yuke Zhu. Lotus: Continual imitation learning for robot manipulation through unsupervised skill discovery. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 537–544. IEEE, 2024.

Xudong Wang, Zebin Han, Zhiyu Liu, Gan Li, Jiahua Dong, Baichen Liu, Lianqing Liu, and Zhi Han. Lifelong language-conditioned robotic manipulation learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18629–18637, 2026.

Maciej Wołczyk, Michał Zajac, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

Iretiayo Akinola Dieter Fox Abhishek Gupta Yashraj Narang Yijie Guo, Bingjie Tang. Srsa: Skill retrieval and adaptation for robotic assembly tasks. In *International Conference on Learning Representations*, 2025.

Amir R Zamir, Alexander Sax, William Shen, Leonidas J Guibas, Jitendra Malik, and Silvio Savarese. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3712–3722, 2018.

Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

A Skill-Relation Feature Construction

A.1 Task-Level Features

Text features use the pretrained Google News Word2Vec model and the in-vocabulary tokens of fixed canonical phrases, yielding 300-dimensional encodings. For in-vocabulary tokens $\mathcal{V}_u$ and $N_2(v) = v/\|v\|_2$ for nonzero v, the task encoding and relation are

$$\begin{aligned} h_u^{\text{text}} &= N_2\left(|\mathcal{V}_u|^{-1} \sum_{v \in \mathcal{V}_u} \text{W2V}(v)\right), \\ R_{\text{text}}(s,t) &= (h_s^{\text{text}})^\top h_t^{\text{text}}. \end{aligned} \tag{10}$$

Discrete motion attributes. The manual task-level code uses 18 binary attributes for the dominant functional operation, covering contact, active and passive object structure, cyclicity, translational and rotational degrees of freedom, and tool use. The code adapts a motion taxonomy to the task level. For binary codes c_s, c_t, its relation is normalized Hamming similarity:

$$R_{\text{attr}}(s,t) = 1 - \frac{d_{\text{H}}(c_s, c_t)}{|c_s|}. \tag{11}$$

Additional visual baseline. We extract ImageNet-pretrained ViT-B/16 CLS features from the first high-view image of each of 50 demonstrations per task. We normalize these 768-dimensional vectors, average them within task, normalize again, and compare tasks by cosine similarity (Table 8).

A.2 Local Motion Distributions

Window measurements. Measurements use active-arm windows of fixed nominal duration. For a window of end-effector poses $(\mathbf{p}_i, R_i)$, define $u_i = \|\mathbf{p}_{i+1} - \mathbf{p}_i\|_2$ and $v_i = \|\delta\boldsymbol{\theta}_i\|_2$, where $\delta\boldsymbol{\theta}_i = \text{Log}(R_i^\top R_{i+1})^\vee$. The implementation normalizes quaternions, makes their signs continuous, and converts consecutive relative quaternions to shortest-angle rotation vectors. The first three descriptors are translational path length $L_{\text{T}} = \sum_i u_i$, rotational path length $L_{\text{R}} = \sum_i v_i$, and translation-to-rotation ratio $r = L_{\text{T}}/L_{\text{R}}$. The composite motion directness index D and translation–rotation correlation C are

$$D = \frac{E_{\text{T}} + E_{\text{R}} + \lambda}{3} - \eta, \qquad C = \text{Corr}(\{u_i\}, \{v_i\}). \tag{12}$$

Here

$$\begin{aligned} E_{\text{T}} &= \frac{\|\mathbf{p}_{n-1} - \mathbf{p}_0\|_2}{L_{\text{T}}}, \quad E_{\text{R}} = \frac{\|\sum_i \delta\boldsymbol{\theta}_i\|_2}{L_{\text{R}}}, \\ \lambda &= \frac{\sigma_1^2}{\sum_k \sigma_k^2}, \qquad\qquad \eta = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathbf{1}[\widehat{\Delta\mathbf{p}_i}^\top \widehat{\Delta\mathbf{p}}_{i+1} < 0]. \end{aligned} \tag{13}$$

where σ_k are the singular values of centered positions and $\mathcal{V}$ contains adjacent pairs with both translation increments nonzero. Zero-path efficiencies, zero-variance λ, and η with no valid pairs are set to zero, using a numerical tolerance of 10^{-12}. Translation–rotation correlation C is signed zero-lag Pearson correlation; fewer than two increments or either standard deviation at most 10^{-12} gives $C = 0$. E_{R} uses the sum of local rotation vectors.

Quantization and gripper categories. The transformed descriptors are

$$\left[\log_{10}(L_{\text{T}} + 10^{-5}),\ \log_{10}(L_{\text{R}} + 10^{-4}),\ \log_{10}(\min(r_{\text{impl}}, 10) + 10^{-4}),\ D,\ C\right], \tag{14}$$

with $r_{\text{impl}} = L_{\text{T}}/L_{\text{R}}$ when $L_{\text{R}} > 10^{-6}$ and $r_{\text{impl}} = 1000$ otherwise. Each of the five transformed measurements is discretized separately into three ordered value ranges. To fit each quantizer, we subtract the median of its fitting values and divide by their interquartile range: the 75th percentile minus the 25th percentile. If this difference is below 10^{-6}, we use a divisor of 1. We then fit one-dimensional k-means with three centers and sort them. The two bin boundaries are the midpoints between adjacent centers, mapped back to the transformed measurement scale and frozen. These

quantizers were fitted on closed-gripper active windows from 24 external AgiBot tasks, with a per-task sampling cap; the same boundaries are applied across gripper categories and tasks. Gripper observations determine open, closed, or transition-or-mixed categories using dataset-specific opening conventions and a pure-state fraction threshold of 0.9. The last category also includes partial opening and mixed windows. These observed-state categories instantiate the context index g in Section 3.1. Gripper icons denote these categories in Fig. 2. These context labels may recur throughout a trajectory.

Aggregation and distributional overlap. The implementation uses five descriptors, three gripper categories, and three bins per descriptor, giving a 45-dimensional local motion distribution. Its conditional-overlap relation is R_{motion}. For valid arm α in demonstration e, with active windows $\mathcal{W}_{e,\alpha}$, the RoboTwin profile is

$$\begin{aligned} z_{e,\alpha}(g,a,b) &= \frac{1}{5|\mathcal{W}_{e,\alpha}|} \sum_{w \in \mathcal{W}_{e,\alpha}} \mathbf{1}[g(w) = g,\ Q_a(w) = b], \\ z_e &= \frac{1}{|\mathcal{A}_e|} \sum_{\alpha \in \mathcal{A}_e} z_{e,\alpha}, \qquad z_u = \frac{1}{|\mathcal{E}_u|} \sum_{e \in \mathcal{E}_u} z_e. \end{aligned} \tag{15}$$

Only arms and demonstrations containing active windows enter the respective averages. Activity requires translation or rotation above its threshold. For the AgiBot task profiles used to fit the shared dictionary (Appendix C.1), active arm windows are pooled within each demonstration before normalization, followed by equal demonstration averaging within each task. Both constructions average demonstrations equally. The factor $1/5$ gives the five features equal weight, so z_e and z_u each have unit total mass. Their coordinates record context–feature marginals. We compare distribution shapes by normalizing each context–feature group (g, a) of the aggregated skill distribution z_u. The conditional distributions and the overlap operator used in Eq. 17 are

$$\begin{aligned} q_u(b \mid g,a) &= \frac{z_u(g,a,b)}{\sum_{b'} z_u(g,a,b')}, \\ \text{Overlap}\big(q_s(\cdot \mid g,a), q_t(\cdot \mid g,a)\big) &= \sum_b \min\{q_s(b \mid g,a), q_t(b \mid g,a)\}. \end{aligned} \tag{16}$$

The first line applies to nonempty groups; a zero-mass group is set to zero. Thus each nonempty three-bin group has unit mass, and a missing group contributes zero even when it is absent from both skills. All $|\mathcal{G}||\mathcal{A}| = 15$ groups remain in the averaging denominator, where $\mathcal{G}$ is the set of gripper categories. Normalization follows demonstration averaging. We average the shared probability mass over all context–feature groups:

$$R_{\text{motion}}(s,t) = \frac{1}{|\mathcal{G}||\mathcal{A}|} \sum_{g,a} \text{Overlap}\big(q_s(\cdot \mid g,a), q_t(\cdot \mid g,a)\big). \tag{17}$$

A.3 Complete-Pose Trajectory Matching

The primary comparison uses both arms in world coordinates, with one temporal alignment shared by the bimanual sequence. The local cost averages the arms' positional distances and quaternion geodesic angles, weighting angles by 0.05 meters per radian. Exact endpoint-constrained DTW uses the symmetrically weighted step pattern: the initial pair and diagonal steps have weight two, and horizontal or vertical steps have weight one. We divide accumulated cost by the sum of sequence lengths and average all 25 pairs of the first five demonstrations per task. Trajectories are neither cropped nor resampled, and gripper commands are excluded. Prespecified variants use start-relative coordinates or positions alone. For fixed complete demonstration subsets $\mathcal{E}_u^{\text{DTW}}$,

$$R_{\text{DTW}}(s,t) = -\frac{1}{|\mathcal{E}_s^{\text{DTW}}||\mathcal{E}_t^{\text{DTW}}|} \sum_{e \in \mathcal{E}_s^{\text{DTW}}} \sum_{f \in \mathcal{E}_t^{\text{DTW}}} d_{\text{DTW}}(\tau_e, \tau_f). \tag{18}$$

B TRANSFER EVALUATION AND REPRESENTATION CONTROLS

B.1 TRANSFER PROTOCOL AND COMPLETE RESULTS

Task suite. We extend RoboTwin with assets and task variants spanning the five behavioral dimensions in Figure 3(a). Fold box uses rigid lids connected by joints as an articulated folding surrogate. The task-level attribute baseline separately encodes each principal operation and functional arm using the 18-bit motion code in Appendix A.

Training and evaluation. Table 2 summarizes the target-adaptation configurations. For $\pi_{0.5}$, source training and target adaptation each use 50 demonstrations, 20,000 updates, and batch size 32. Source training updates the full policy; target adaptation freezes the source-specialized language backbone and updates the action side. Both stages use the same optimizer and learning-rate schedule. The cosine schedule has a configured duration of 30,000 updates and a terminal rate of 2.5×10^{-6}. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.95)$, $\epsilon = 10^{-8}$, and weight decay 10^{-10}, with global-gradient-norm clipping at 1.0 and EMA decay 0.99. Each adaptation loads the source policy weights into a new training run. Each adapted policy is evaluated on 50 episodes. Target-only baselines start directly from the common initialization and use the same target data, adaptation settings, trainable-module restriction, update budget, and evaluation condition.

For ACT, we split the 50 demonstrations into 40 training and 10 validation episodes. Each epoch samples one starting point per training episode, giving five minibatches at batch size 8. Source training uses 6,000 epochs (30,000 optimizer updates), and adaptation uses 4,000 epochs (20,000 optimizer updates). We initialize adaptation from the source's minimum-validation-loss checkpoint with a fresh optimizer and select the adapted checkpoint by target validation loss. ACT trains its policy, including the ResNet-18 backbone, with learning rate 10^{-5} for both backbone and other parameters. Its transformer hidden dimension is 512, feed-forward dimension is 3,200, and the latent KL-loss weight is 10. AdamW uses $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and weight decay 10^{-4}, with a constant learning rate and no EMA.

For GR00T-N1.7, source training is configured for 20,000 updates and target adaptation for 10,000 updates. The reported transferability measurements use the target checkpoint at 5,000 updates. Both stages freeze the language and vision backbones while training the action-side projectors, vision-language normalization, and diffusion transformer. Adaptation loads the corresponding source checkpoint and uses AdamW with learning rate 10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 10^{-5}, and global-gradient-norm clipping at 1.0, without EMA. Cosine decay follows a warmup of 5% of the configured training budget, giving 500 warmup updates for adaptation. Each target training sample supervises 16 action steps; the processor pads this sequence to 40 steps and masks the padded positions out of the loss.

For StarVLA, each adaptation independently loads its source checkpoint, resets the optimizer, and trains only the action diffusion transformer. The Qwen vision-language backbone, projectors, and other action-head components remain frozen. Training uses one GPU with batch size 64 and no gradient accumulation, 20,000 updates, and 16-step action sequences. The effective run configuration uses fused AdamW with peak learning rate 10^{-4} and 100 warmup updates.

Evaluation panels and observed outcomes. The main representation comparisons use the $\pi_{0.5}$ panel of 144 directed edges: six candidate sources for each of 24 targets, 39 distinct sources, and a union of 42 tasks. ACT and GR00T-N1.7-3B use the same 36 source–target pairs, with four target-specific sources for each of nine targets. These sources were selected using existing $\pi_{0.5}$ measurements and manipulation-structure coverage, with 15 distinct source tasks across the 36 pairs. StarVLA (QwenGR00T with robot state inputs) uses the same nine targets and available source checkpoints, with 36 observed pairs and four sources per target; its source selections differ from those of ACT and GR00T-N1.7-3B to further test whether the relation's agreement with transfer rankings generalizes across candidate-source sets. Each observed pair for these three policies is evaluated on 25 episodes. For each policy, we compute ordinary within-target Spearman correlations between the fixed motion relation and source-conditioned target success, then average the defined target correlations.

Table 2: Target-adaptation settings for transferability measurement. Configured training budgets and evaluated checkpoints are listed separately; batch sizes are global.

Parameter	$\pi_{0.5}$	ACT	GR00T-N1.7	StarVLA
Training configuration				
Target demonstrations	50	50	50	50
Batch size	32	8	64	64
Configured update budget	20,000	20,000	10,000	20,000
Evaluated checkpoint	20,000	Best validation	5,000	20,000
Optimization				
Optimizer	AdamW	AdamW	AdamW	Fused AdamW
Peak learning rate	2.5×10^{-5}	10^{-5}	10^{-4}	10^{-4}
Warmup updates	1,000	0	500	100
Action supervision				
Supervised action steps	16	50	16	16

The complete $\pi_{0.5}$ transfer gains are in Table 3; the ACT, GR00T-N1.7-3B, and StarVLA source-conditioned success rates are in Tables 4, 5, and 6, respectively. Task abbreviations are defined in Table 7.

Table 3: All 144 transfer gains (percentage points), with target-only success rates (Base, %). Each row lists its six sources in decreasing gain; bold source labels mark all maxima.

Target	Base (%)	Evaluated sources and gains (pp), in decreasing gain order					
Adjust bottle	90	**Roller** +10	**Bottl** +10	**ShakeH** +10	Lift +8	Dual +8	Knob −86
Hammer block	0	**Stamp** +86	Cut +74	Roll +68	Rank +64	Staple +50	Push +38
Close microwave	92	**Rotate** +4	Lift 0	Open −6	Fold −10	Hang −12	Knob −80
Cut bread	94	**Hammer** +6	**Push** +6	**Roll** +6	**Stamp** +6	**Wipe** +6	Staple +4
Drag waterer	92	**Roll** −4	**Wipe** −4	Card −8	CanPot −10	Sweep −12	Push −14
Erase board	48	**Wipe** +30	Drag +20	Sweep +12	Hammer +2	Roll +2	Push 0
Fold box	12	**Hang** +14	**Bowls** +14	Close +10	Stk3 +4	Stk2 0	Open −8
Handover block	12	**Dual** +32	Hang +24	Stk2 +18	Fold +10	Basket +6	CanPot −6
Hang mug	14	**Fan** +16	Rotate +12	Left +4	Fold −10	Open −12	Close −14
Lift pot	72	**CanPot** +20	Dual +18	Close +16	Hang +16	Laptop +16	Bottl +10
Move pill bottle	62	**CanPot** +16	Cans −6	Card −14	Roll −24	Right −26	Push −46
Open microwave	0	**Close** +32	Drag +14	Laptop +10	Rotate +10	Fold +4	Hang +2
Press stapler	88	**Alarm** −8	Hammer −10	Bell −12	Push −12	Stamp −12	Cut −14
Push blocks	96	**Rank** +4	**CanPot** +4	Hammer +2	Cut +2	Drag +2	Roll +2
Roll cards	12	**Card** +36	Sweep +26	Drag +6	Rotate +4	Wipe +4	Push −6
Rotate QR code	58	**Hang** +12	ShakeH +2	Roll −16	Open −20	Switch −26	Knob −56
Scan object	32	**Roll** +18	Rotate +6	Stamp +6	Staple −2	Dual −8	Knob −32
Shake bottle	92	**Lift** +4	**Bottl** +4	Adj +2	Rotate +2	ShakeH +2	Knob −60
Stack bowls	32	**Stk2** +46	Left +38	Stk3 +38	Cans +24	Fold +22	Lift 0
Stamp seal	44	**Hammer** +12	**Cut** +12	Bell +10	Scan +8	Staple +4	Switch −12
Stir bowl	8	**Hammer** +20	Cut +14	Sweep +14	Stamp +12	Rotate +10	Knob −8
Sweep floor	66	**Drag** +18	Erase +14	Wipe +2	Push −2	Roll −22	Stir −26
Turn MW knob	18	**Stamp** +20	Stir +2	Close −2	Hang −4	Open −6	Rotate −8
Wipe table	2	**Stir** +40	Cut +24	Erase +22	Sweep +18	Drag +16	Roll +14

The main representation comparisons in Section 3 have finite scores for every edge in this panel. Transfer correlations use the within-target Spearman criterion in Eq. 4, with 24 defined targets for all main methods. Correlations require at least three sources and nonconstant relation and outcome ranks. Outcome ranks are permuted within each target, and exact ties receive average ranks.

Additional representation baselines. Table 8 adds a first-frame visual baseline to the main representations, using the same 144-edge panel and ordinary Spearman criterion.

The visual and motion-distribution pipelines use 50 demonstrations for each of the 43 tasks. The DTW pipeline uses the first five demonstrations in numeric episode order, totaling 215 complete sequences, from the available clean and randomized demonstrations. Its outcome endpoint is clean target success. DTW cost and alignment choices are specified in Appendix A. The prespecified start-relative pose variant yields the same fixed-panel correlation as world-pose DTW (-0.094), while the position-only variants yield -0.125. The normalized distances are numerically different but induce the same within-target rankings in this collection.

Table 4: **ACT.** Source-conditioned target success (36 pairs).

	Candidate 1		Candidate 2		Candidate 3		Candidate 4	
Target	Source	%	Source	%	Source	%	Source	%
Close microwave	Fold	**68**	Knob	56	Lift	40	Rotate	16
Erase board	Staple	**12**	Drag	0	Roll	0	Wipe	0
Handover block	Fold	**80**	CanPot	76	Stk2	60	Dual	56
Hang mug	Fan	**12**	Left	**12**	Fold	4	Rotate	0
Open microwave	Drag	**8**	Fold	0	Knob	0	Lift	0
Push blocks	Drag	**56**	CanPot	52	Roll	28	Hammer	0
Scan object	Rotate	**12**	Dual	0	Knob	0	Roll	0
Shake bottle	ShakeH	**84**	Knob	80	Lift	72	Rotate	52
Stack bowls	Stk2	**84**	Lift	80	Fold	64	Left	40

Success (%), decreasing within each row; bold marks maxima. Abbreviations: Table 7.

Table 5: **GR00T-N1.7-3B.** Source-conditioned target success (36 pairs).

	Candidate 1		Candidate 2		Candidate 3		Candidate 4	
Target	Source	%	Source	%	Source	%	Source	%
Close microwave	Lift	**64**	Fold	56	Knob	44	Rotate	28
Erase board	Wipe	**12**	Roll	8	Drag	0	Staple	0
Handover block	CanPot	**0**	Dual	**0**	Fold	**0**	Stk2	**0**
Hang mug	Fan	**0**	Fold	**0**	Left	**0**	Rotate	**0**
Open microwave	Fold	**36**	Drag	24	Lift	16	Knob	4
Push blocks	Roll	**80**	Drag	76	CanPot	72	Hammer	68
Scan object	Rotate	**4**	Dual	0	Knob	0	Roll	0
Shake bottle	Knob	**100**	ShakeH	96	Lift	88	Rotate	88
Stack bowls	Stk2	**24**	Fold	20	Left	8	Lift	4

Success (%), decreasing within each row; bold marks maxima. Abbreviations: Table 7.

Table 6: **StarVLA.** Source-conditioned target success (36 pairs).

	Candidate 1		Candidate 2		Candidate 3		Candidate 4	
Target	Source	%	Source	%	Source	%	Source	%
Close microwave	Lift	**56**	Fold	40	Open	8	Rotate	8
Erase board	Cut	**12**	Drag	4	Push	4	Staple	4
Handover block	Bowls	**0**	Fold	**0**	Lift	**0**	Pill	**0**
Hang mug	Close	**0**	Fold	**0**	Open	**0**	Rotate	**0**
Open microwave	Close	**4**	Drag	0	Fold	0	Lift	0
Push blocks	Cut	**52**	Pill	36	Drag	28	Staple	0
Scan object	Rotate	**4**	Staple	**4**	Lift	0	Stamp	0
Shake bottle	Stir	**60**	Adj	16	Lift	12	Rotate	0
Stack bowls	Pill	**40**	Fold	4	Lift	4	Rotate	0

Success (%), decreasing within each row; bold marks maxima. Abbreviations: Table 7.

Table 7: Task abbreviations and example scenes used in the source–target outcome tables. Each entry gives the abbreviation, canonical task identifier, and the first high-view frame of a demonstration.

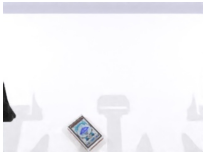
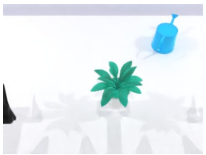
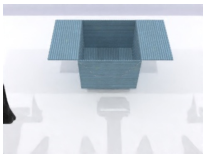
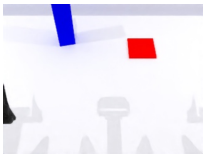
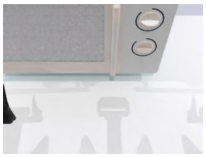
Adj adjust_bottle	**Alarm** click_alarmclock	**Basket** place_object_basket
		
Bell click_bell	**Bottl** pick_diverse_bottles	**Bowls** stack_bowls_two
		
CanPot move_can_pot	**Cans** place_cans_plasticbox	**Card** move_playingcard_away
		
Close close_microwave	**Cut** cut_bread	**Drag** drag_waterer
		
Dual pick_dual_bottles	**Erase** erase_board	**Fan** place_fan
		
Fold fold_box	**Hammer** beat_block_hammer	**Hando** handover_block
		
Hang hanging_mug	**Knob** turn_microwave_knob	**Laptop** open_laptop
		

Table 7 (continued): Task abbreviations and example scenes.

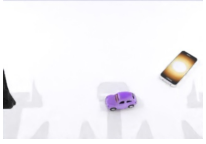
Left place_a2b_left	**Lift** lift_pot	**Open** open_microwave
		
Pill move_pillbottle_pad	**Push** push_blocks_hit	**Rank** blocks_ranking_size
		
Right place_a2b_right	**Roll** roll_playingcards	**Roller** grab_roller
		
Rotate rotate_qrcode	**Scan** scan_object	**Shake** shake_bottle
		
ShakeH shake_bottle_horizontally	**Stamp** stamp_seal	**Staple** press_stapler
		
Stir stir_bowl	**Stk2** stack_blocks_two	**Stk3** stack_blocks_three
		
Sweep sweep_floor_with_broom	**Switch** turn_switch	**Wipe** wipe_table_with_brush
		

Table 8: Representation baselines, including an additional first-frame visual baseline, on the retrospective 144-edge panel. Entries are standard Spearman correlations averaged equally across the 24 targets with defined values for every method. Intervals are 95% target-bootstrap intervals.

Description	$\overline{\rho}$	95% interval
Canonical task phrase	0.067	[-0.126, 0.253]
Manual motion attributes	0.121	[-0.058, 0.299]
Motion distributions	0.376	[0.185, 0.551]
Complete-pose DTW	-0.094	[-0.277, 0.087]
First-frame vision (additional)	0.161	[0.005, 0.311]

Table 9: Task-level motion attributes on the 144-edge panel using ordinary Spearman. Intervals are 95% target-bootstrap intervals. The five attribute blocks form one q-value adjustment family; the complete code is excluded.

Attribute block	Targets	$\overline{\rho}$	95% interval	q
Complete code	24	0.121	[-0.058, 0.299]	–
Contact	14	0.039	[-0.192, 0.268]	0.836
Structure	12	-0.145	[-0.371, 0.104]	0.437
Active trajectory	23	-0.020	[-0.206, 0.161]	0.836
Passive trajectory	24	0.163	[-0.033, 0.358]	0.191
Tool use	13	0.306	[0.039, 0.543]	0.064

Attribute-level support and manual coding. The motion taxonomy is adapted manually to a task's principal operation and functional arm; preparatory approach and grasping are excluded from that task-level code. All 43 task codes are present. We report each block over targets with nonconstant scores in Table 9.

B.2 Local Motion Representation Ablations

We ablate the local-motion representation to examine the respective contributions of motion characteristics and gripper context to its transfer relevance. Under the same evaluation protocol, we compare the full representation against two controls: removing gripper conditioning or five motion descriptors. All variants use the same task profiles and clean target success rates. We report ordinary within-target Spearman correlations averaged equally over the 24 targets in Table 10.

Contribution of gripper conditioning. To isolate the role of gripper context, we sum z_u over the three gripper categories and normalize the three-bin distribution of each motion descriptor. The resulting relation averages histogram intersection across the five descriptor distributions and achieves $\overline{\rho} = 0.280$, compared with 0.376 for the full representation. This reduction indicates that conditioning local motion on gripper state captures interaction context that is informative for distinguishing transfer-relevant skill relationships.

Contribution of motion descriptors. We next remove the five motion descriptors and retain only the marginal frequencies of the three gripper states, obtained by summing z_u over descriptors and motion bins. Comparing these three-dimensional vectors by histogram intersection yields only $\overline{\rho} = 0.161$. Thus, gripper-state usage alone provides limited transfer-relevant information, whereas the motion descriptors account for a substantial part of the observed correspondence. Together, these ablations suggest that local motion structure provides the primary signal for characterizing transfer-relevant skill relationships, while gripper conditioning supplies complementary interaction context. Their joint representation yields better correspondence with measured transfer.

Table 10: Ablations on components used in aggregating local motion distributions. Correlations are averaged equally over 24 targets; intervals are 95% target-bootstrap intervals.

Variant	$\overline{\rho}$	95% interval
Full local motion distributions	0.376	[0.185, 0.551]
w/o gripper conditioning	0.280	[0.076, 0.475]
w/o motion descriptors	0.161	[-0.026, 0.338]

B.3 SOURCE-RANKING DIAGNOSTICS

For Figure 4(c), each of the 24 targets has six candidate sources (144 transfers). Comparing every two sources gives $\binom{6}{2} = 15$ comparisons per target, or $24 \times 15 = 360$ in total. We exclude 40 comparisons with identical observed transfer gains because they provide no source ordering to evaluate. The remaining 320 comparisons cover all 24 targets. For target t, $\mathcal{P}_t$ contains its source pairs with unequal gains. For relation R, $N_{\mathrm{tie,t}}(R)$ counts score ties in $\mathcal{P}_t$ and $N_{\mathrm{inv,t}}(R)$ counts strictly opposite score and gain orderings:

$$\begin{aligned} L_{\mathrm{tie}}(R) &= \frac{1}{|\mathcal{T}_{\mathrm{pair}}|} \sum_{t \in \mathcal{T}_{\mathrm{pair}}} \frac{N_{\mathrm{tie,t}}(R)}{2|\mathcal{P}_t|}, \\ L_{\mathrm{inv}}(R) &= \frac{1}{|\mathcal{T}_{\mathrm{pair}}|} \sum_{t \in \mathcal{T}_{\mathrm{pair}}} \frac{N_{\mathrm{inv,t}}(R)}{|\mathcal{P}_t|}, \qquad L(R) = L_{\mathrm{tie}}(R) + L_{\mathrm{inv}}(R). \end{aligned} \tag{19}$$

Here $\mathcal{T}_{\mathrm{pair}} = \{t : |\mathcal{P}_t| > 0\}$. Eligibility requires $|G_i - G_j| > 10^{-12}$, equivalent to thresholding $|T_i - T_j|$ because the target-only baseline is shared. Score ties group sorted values within 10^{-12} of the group's smallest value. Both components use the same denominator: a constant relation has zero inversion loss but total loss $1/2$.

Attributes use the full 18-bit Hamming similarity. DTW averages the 25 pairs of five-demonstration pose sequences per task pair; local statistics use the same demonstrations, phase-conditioned overlap, frozen encoding, and equal-demonstration aggregation. The 95% percentile intervals use 20,000 paired target-bootstrap draws shared across methods and loss components. Start-aligned and world-frame DTW give the same losses. Weighting source pairs by their absolute gain gaps, normalized within target, gives total losses of 45.2%, 56.8%, and 27.6% for attributes, DTW, and local statistics, respectively.

Transfer gains from modeling of local motion distributions. Here we provide analysis on the practical transfer gain using best source candidates matched with local motion distribution similarities. Let $T_{s,t}$ be observed success after adapting source s to target t, and b_t the target-only success rate under the same evaluation condition. The transfer gain in percentage points and the selected source are

$$G_{s,t} = 100(T_{s,t} - b_t), \qquad s_t^* = \arg\max_{s \in \mathcal{S}_t} R_{\mathrm{motion}}(s, t). \tag{20}$$

Within each fixed candidate set, the Top-1 decision uses only the score estimated by relationships on local motion distributions. All 24 targets have a unique highest-scoring source. We compare this Top-1 rule with selecting one source uniformly from all source candidates and with the best observed candidate, giving

$$\begin{aligned} \overline{G}_{\mathrm{top1}} &= \frac{1}{24} \sum_t G_{s_t^*,t}, \qquad \overline{G}_{\mathrm{uniform}} = \frac{1}{24} \sum_t \frac{1}{6} \sum_{s \in \mathcal{S}_t} G_{s,t}, \\ \overline{G}_{\mathrm{best}} &= \frac{1}{24} \sum_t \max_{s \in \mathcal{S}_t} G_{s,t}. \end{aligned} \tag{21}$$

Uniform selection is evaluated by its exact expectation rather than one random draw. The best observed candidate uses the measured outcomes and provides an oracle reference within the six evaluated sources. For each rule, regret is the target-averaged gap to this reference; the negative-gain frequency averages the probability of selecting a source with $G_{s,t} < 0$ under that rule. The result is reported in Table 11

Table 11: Retrospective source selection on the same 24-target, 144-edge panel. Gain is relative to target-only adaptation. Intervals use paired target bootstrap. Neg. is the frequency of an observed negative gain; regret is the gap to the best observed candidate.

Selection rule	Gain (pp)	95% interval	Neg. (%)	Regret (pp)
Top-1 by R_{motion}	13.58	$[7.25, 21.83]$	12.50	6.58
Uniform over six sources	4.90	$[-1.39, 12.29]$	31.25	15.26
Best observed source	20.17	$[13.17, 28.42]$	8.33	0.00

C Shared Behavioral Primitives and Skill Compositions

C.1 Dictionary Fitting and Projection

Factorization and basis size. We fit the 188 external AgiBot task profiles as $X \approx WH$ with nonnegative factors and normalize each basis row to unit mass. Projection weights are nonnegative without a unit-sum constraint, so the objective is generalized KL divergence:

$$D_{\mathrm{KL}}(x\|y) = \sum_j \left[x_j \log \frac{x_j}{y_j} - x_j + y_j \right]. \tag{22}$$

Zero-mass log terms contribute zero; positive mass against a zero prediction gives infinite divergence. The previously selected $K = 6$ is retained for the 50K fit. Its selection used 3,743 valid demonstrations from the same tasks, without transfer outcomes. Candidates $K = 2, \ldots, 8$ were evaluated by held-out task reconstruction (JS divergence) and task-bootstrap component alignment (cosine). The rule chooses the smallest K within one standard error of the best reconstruction whose weakest component has mean matched cosine at least 0.9. If none meets both criteria, it chooses the best reconstruction among candidates meeting the stability criterion; $K = 6$ follows this fallback.

50K sampling and fitting. The expanded sample contains 50,000 distinct episodes from the same task corpus; 49,973 have active sampled windows. Eligible historical episodes retain their action chunks. Additional episodes are selected in fixed hash order within tasks and round-robin order across tasks; a fixed hash selects one action chunk per episode, from which we take the central 30 frames. Reserved validation episodes are excluded, and the original quantizers remain frozen. Active arm windows are pooled and normalized within each episode, valid episodes are averaged equally within task, and all task profiles receive equal fitting weight. We compare three initializations by training KL error and retain the lowest-error fit. The resulting H is frozen for the main map, composition validation, and transfer evaluation; component interpretations come from this dictionary's own feature distributions.

Demonstration projection. Each evaluation demonstration is projected separately and its weights are normalized before task averaging:

$$\begin{aligned} w_e &\approx \arg\min_{w \geq 0} D_{\mathrm{KL}}(z_e \| wH), & \widetilde{w}_e &= w_e / \|w_e\|_1, \\ m_u &= |\mathcal{E}_u|^{-1} \sum_{e \in \mathcal{E}_u} \widetilde{w}_e, & R_{\mathrm{basis}}(s,t) &= \frac{m_s^\top m_t}{\|m_s\|_2 \|m_t\|_2}. \end{aligned} \tag{23}$$

These are soft compositions over six components. The input retains gripper-state mass. All expanded-dictionary projections use 2,000 fixed iterations per demonstration, independent of other demonstrations in a batch.

C.2 Visualizing Skill Kinship through Shared Primitives

Kinship visualization and held-out skills. We fit the frozen six-component dictionary using a 50,000-episode sample from the same task corpus (Appendix C.1). For validation, five tasks from the fitted corpus contribute held-out episodes, while seven additional tasks are excluded from dictionary and quantizer fitting. Each of these 12 tasks contributes 50 complete demonstrations (Appendix C). We construct compositions for nine skill labels by re-extracting 11,294 saved 30-frame windows from 1,820 episodes with the frozen feature construction and quantizers. Normalized projection weights are averaged successively within segments, episodes, tasks, and labels, giving each child unit equal weight at each level. Figure 5(b) shows five of these labels and one held-out task. Rings retain all six weights; each node connects to at most its two largest components with weights of at least 20%, and line width reflects composition weight. Dashed links mark the held-out example. Pour, Scoop, and Wipe illustrate different contributions of shared components; Insert combines M0 with M5. Cap, a held-out task involving capping and storing markers, averages compositions from 50 complete demonstrations and shares the M1/M5 pair with Grasp.

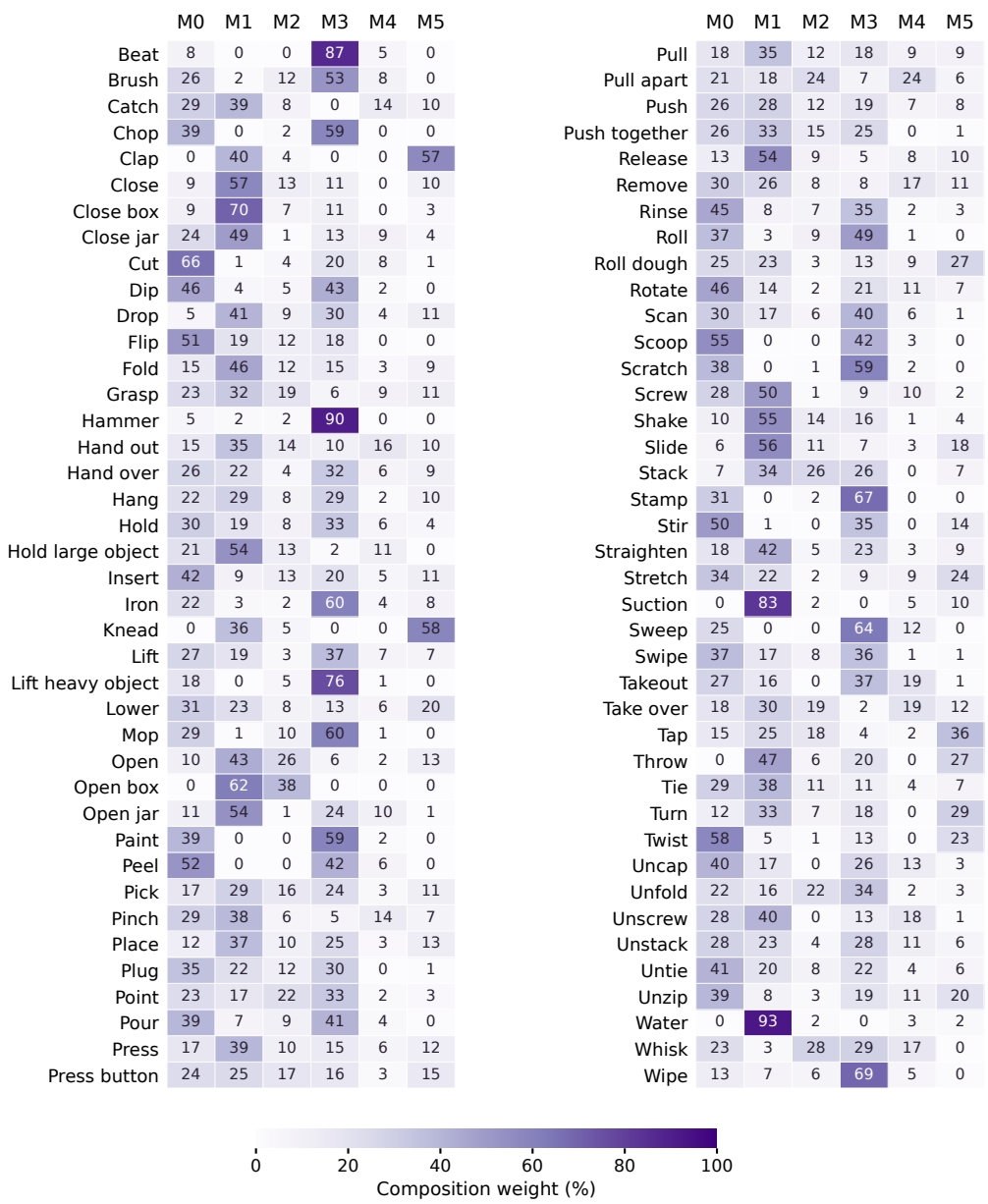


Figure 8: Complete manipulation-skill inventory under the original frozen six-component dictionary. All 80 skill labels are arranged alphabetically across two panels, with 40 labels per panel. Columns denote shared primitives; cell values give task-balanced composition weights in percent.

Complete skill inventory. We present all 80 manipulation skill labels from the expanded labeled corpus in Figure 8, using the original frozen six-component dictionary. The compositions average normalized window weights within segments, episodes, and tasks, followed by an equal-weight average across tasks for each label. The inventory includes skills such as folding, cutting, stirring, fastening, and object handover.

C.3 Primitive and Composition Validation

The main evaluations below use the same frozen six-component dictionary fitted from 50,000 distinct sampled episodes (Appendix C.1).

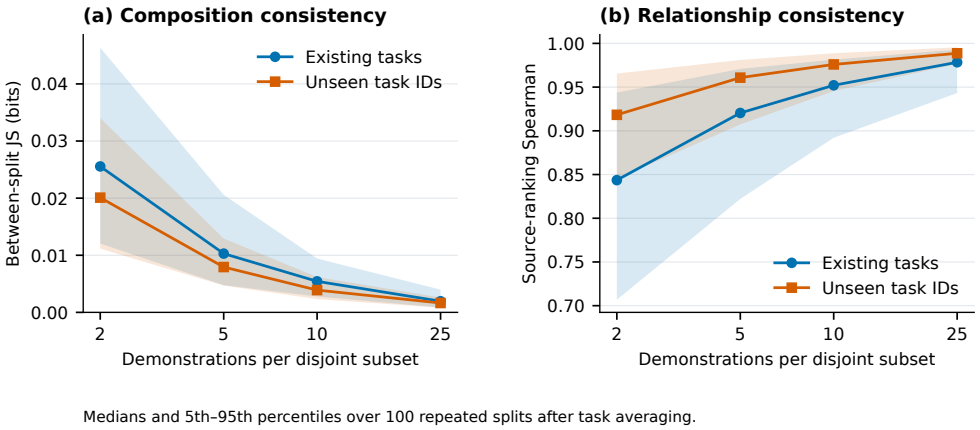


Figure 9: Composition and relation stability with the frozen 50K dictionary. Both groups use 50 held-out complete recordings per task. Lines show medians after task averaging; bands show fifth–ninety-fifth percentiles over repeated disjoint splits.

Primitive interpretation and resampling. We summarize each primitive by its three-bin marginals for each motion descriptor and its gripper-state mass. Projection uses the full 45-entry rows. For example, M1 and M3 have open and closed gripper mass 0.772 and 0.972, respectively, while their high-directness bin masses are 0.917 and 0.858. M2 and M4 place 0.832 and 0.937 on the low-translation bin. M5's transition-or-mixed gripper mass is 0.159, compared with 0.004 for M0.

Each of 30 bootstrap repetitions draws, with replacement, a sample of the same size as the fitting task-profile set, fits three initializations, selects by training KL, and uses a one-to-one Hungarian match to maximize total cosine with the main H. The episode estimates within task are held fixed. Median matched cosines for M0–M5 are $0.958, 0.987, 0.990, 0.995, 0.996, 0.953$; their fifth percentiles are $0.866, 0.976, 0.967, 0.987, 0.987, 0.581$. All selected fits converge before the 2,000-iteration limit. M1–M4 are consistently recovered across resamples, whereas M5 is more sensitive to the fitting task mixture despite its high median alignment.

Reproducibility across demonstration subsets. We test whether different demonstrations of the same task yield consistent primitive compositions and similar rankings of related tasks (Figure 9). We evaluate five tasks from the fitting corpus using held-out demonstrations, and seven new tasks excluded from both dictionary and quantizer fitting. Each task contributes 50 demonstrations, giving 600 episodes with no overlap with the 50K fitting set. Demonstrations for the five existing tasks come from the separate 1K validation corpus.

For validation, we aggregate 30-frame windows at stride 15 throughout each complete recording, using the same pose and gripper channels, frozen quantizers, and activity thresholds. Active windows are averaged within arm, followed by equal valid-arm averaging. We then project and normalize each demonstration as in Eq. 23. In each of 100 repetitions, we randomly split each task's 50 demonstrations into two disjoint sets of 25. From each set, nested subsets of $n = 2, 5, 10, 25$ demonstrations independently estimate the task composition. We compare the two compositions by JS divergence in bits (lower is more consistent), and their similarity rankings over a fixed reference library by Spearman correlation (higher is more consistent). The library includes an independently estimated same-task reference for existing tasks. We average within each task group before reporting medians and fifth–ninety-fifth percentiles across the repeated splits.

We present the stability curves in Figure 9. At $n = 25$, median task-mean composition JS is 0.00195 for existing tasks and 0.00164 for unseen tasks, with relation-ranking agreements of 0.978 and 0.989, respectively.

Full demonstrations versus single windows. To test whether aggregating the full demonstration helps reproducibility, we repeat the preceding comparison using only one 30-frame window per episode, keeping the same 600 episodes, dictionary, and splits. At 25 demonstrations per subset, composition JS rises to 0.05797 and 0.04885, while ranking agreement falls to 0.7664 and 0.7809, for existing and unseen tasks, respectively. Thus, aggregating windows throughout each demonstration gives more consistent compositions and similarity rankings across demonstration subsets.

Transfer correspondence. We independently project 2,150 RoboTwin demonstration distributions (43 tasks, 50 demonstrations each) onto the frozen 50K dictionary and average normalized coefficients within tasks. Composition cosine is compared with the original local-distribution relation on exactly the same 144 source–target edges, six sources for each of 24 targets. This compositional analysis uses a four-percentage-point outcome tolerance; Section 3 uses ordinary outcome ranks. For each target t, define

$$d^{\delta}_{t,i} = \sum_{j=1}^{6} \mathbf{1}[T_{t,i} > T_{t,j} + \delta]\,, \qquad \delta = 0.04, \quad \overline{\rho}_{\delta} = \frac{1}{|\mathcal{T}_{\delta}|} \sum_{t \in \mathcal{T}_{\delta}} \mathrm{Spearman}(\mathbf{r}_t, \mathbf{d}^{\delta}_t). \tag{24}$$

Thus, $d^{\delta}_{t,i}$ counts sources whose success rates are exceeded by more than the tolerance. Tied scores receive average ranks. All three relations below have defined correlations on the same 22 targets, which receive equal weight. Cut bread and Push blocks hit have constant dominance counts and are excluded. The implementation uses a 10^{-12} numerical guard at the tolerance boundary and the same precision for score ties. Confidence intervals use 20,000 target-bootstrap resamples.

Combining distributional and compositional relations. We combine composition cosine R_{basis} and the original local-distribution overlap R_{motion} by equal-weight rank fusion. For each target, let U_t map scores over all 42 non-self sources in the fixed 43-task library to unit midranks: $U_t(\mathbf{x}) = 2\,\mathrm{rank}_0(\mathbf{x})/41 - 1$, where rank_0 uses zero-based average ranks. The fused scores are

$$\mathbf{r}^{\mathrm{fuse}}_t = U_t\left(\frac{U_t(\mathbf{r}^{\mathrm{basis}}_t) + U_t(\mathbf{r}^{\mathrm{motion}}_t)}{2}\right). \tag{25}$$

We then select the six evaluated sources per target for the transfer comparison. Table 12 shows that composition cosine correlates positively with measured transfer; rank fusion has a higher point estimate than either relation alone.

Table 12: Transfer correspondence with a four-percentage-point outcome tolerance. All relations use the same 22 valid targets; 95% intervals resample targets.

Relation	$\overline{\rho}_{\delta}$	95% interval
50K composition	0.372	$[0.180, 0.541]$
Local distribution	0.429	$[0.227, 0.608]$
Rank fusion	0.480	$[0.286, 0.646]$

D CONTINUAL-LEARNING IMPLEMENTATION DETAILS

Composition estimation and source selection. For each demonstration $e \in \mathcal{E}_t$ of the arriving skill, we construct its local-motion distribution z_e as in Appendix A. The quantizers and behavioral basis H ($K = 6$) remain fixed throughout the continual stream. Following Eq. 7, we obtain non-negative coefficients w_e by minimizing $D_{\mathrm{KL}}(z_e \| w_e H)$, normalize each vector as $\widetilde{w}_e = w_e / \|w_e\|_1$, and compute $m_t = |\mathcal{E}_t|^{-1} \sum_{e \in \mathcal{E}_t} \widetilde{w}_e$. Thus, demonstrations contribute equally after individual projection and normalization. For $t > 1$, we compare m_t with stored compositions m_s, $s < t$, using cosine similarity $R_{\mathrm{basis}}(s, t)$ (Eq. 6) and select a maximizing source d_t as in Eq. 8.

Adapter initialization. At each of $L = 18$ prefix FFN layers, the adapter is a bottleneck MLP with a down-projection, ReLU, and up-projection. As in Eq. 8, the selected source supplies the new branch's initial weights and biases at every adapted layer. These parameters form $\Phi_t^{(0)}$; the first skill uses a fresh adapter. Source and target branches have independent parameters. We freeze the backbone and all historical branches, initialize a fresh optimizer, and train only the new adapter on current demonstrations $\mathcal{D}_t$.

Policy training. The flow-matching loss $\mathcal{L}_{\mathrm{FM}}$ in Eq. 9 is

$$\mathcal{L}_{\mathrm{FM}}(\theta_0, \Phi; \mathbf{o}, \mathbf{a}) = \mathbb{E}_{\boldsymbol{\epsilon}, \tau} \left[\| v_{\theta_0, \Phi}(\mathbf{o}, \mathbf{x}_\tau, \tau) - (\boldsymbol{\epsilon} - \mathbf{a}) \|_2^2 \right], \tag{26}$$

where $\mathbf{x}_\tau = \tau \boldsymbol{\epsilon} + (1 - \tau)\mathbf{a}$, $\mathbf{a}$ is a demonstrated action sequence, $\boldsymbol{\epsilon}$ is Gaussian noise, $\tau \in [0, 1]$ is the sampled noise level, and θ_0 denotes the frozen backbone. Optimization starts from $\Phi_t^{(0)}$ and updates only Φ_t, yielding $\Phi_t^\star$.

Simulation training configuration. Table 13 gives the training settings shared by relation-guided and random source initialization. The policy predicts 16-step action sequences. For each incoming task, we use batch size 64 for 5,000 adapter updates, followed by 2,000 reconstruction-autoencoder updates. Both phases use constant learning rates and fresh AdamW states, with no warmup or EMA. After training, we collect routing statistics over 1,000 additional batches without gradient updates. Inference uses the raw reconstruction errors defined below.

Table 13: Simulation continual-learning settings for relation-guided and random source initialization. Each task has separate adapter and reconstruction-autoencoder training phases.

Parameter	Adapter	Reconstruction autoencoder
Training configuration		
Batch size	64	64
Updates per task	5,000	2,000
Trainable parameters	Current-task adapter	Current-task autoencoders
Optimization		
Optimizer	AdamW	AdamW
Learning rate	10^{-4}	5×10^{-4}
Schedule	Constant	Constant
Adam (β_1, β_2)	$(0.9, 0.999)$	$(0.9, 0.999)$
Adam ϵ	10^{-8}	10^{-8}
Weight decay	10^{-10}	10^{-2}
EMA	None	None
Architecture		
Prefix FFN layers	18	18
Hidden width	128	128
Autoencoder latent dimension	—	64

Reconstruction autoencoder training. After adapter training, we freeze the policy and fit a fresh autoencoder $\psi_t^{(\ell)}$ at each adapted layer using only current-task observations. Let $\mathbf{h}_j^{(\ell)}(\mathbf{o}) \in \mathbb{R}^{d_\ell}$

be the normalized pre-FFN prefix features, $\mathcal{M}(\mathbf{o})$ the valid prefix-token indices, and sg the stop-gradient operator. For $\Psi_t = \{\psi_t^{(\ell)}\}_{\ell=1}^{L}$, the reconstruction error and training objective are

$$e_t^{(\ell)}(\mathbf{o}) = \frac{1}{d_\ell |\mathcal{M}(\mathbf{o})|} \sum_{j \in \mathcal{M}(\mathbf{o})} \left\| \psi_t^{(\ell)}\left(\mathrm{sg}[\mathbf{h}_j^{(\ell)}(\mathbf{o})]\right) - \mathrm{sg}[\mathbf{h}_j^{(\ell)}(\mathbf{o})] \right\|_2^2,$$
$$\Psi_t^\star = \arg\min_{\Psi_t} \mathbb{E}_{\mathbf{o}\sim\mathcal{D}_t}\left[\frac{1}{L}\sum_{\ell=1}^{L} e_t^{(\ell)}(\mathbf{o})\right]. \tag{27}$$

We expand the skill library with $(m_t, \Phi_t^\star, \Psi_t^\star)$: the composition supports subsequent source selection, while the learned autoencoders support inference routing.

Inference routing. Each adapted layer selects the skill with the lowest raw reconstruction error and uses its adapter:

$$\widehat{k}_\ell(\mathbf{o}) = \arg\min_{1\le k\le t} e_k^{(\ell)}(\mathbf{o}),$$
$$\mathrm{FFN}_\ell^{\mathrm{CL}}(\mathbf{h}) = \mathrm{FFN}_\ell^{0}(\mathbf{h}) + A_{\widehat{k}_\ell(\mathbf{o}),\ell}(\mathbf{h}). \tag{28}$$

Here $A_{k,\ell}$ is skill k's adapter at layer ℓ. Different layers may select different skills. Composition similarity guides initialization; reconstruction error controls inference routing.

E CONTINUAL-LEARNING EVALUATION

E.1 SIMULATION PROTOCOL AND RESULTS

Table 14: Simulation metrics on the ten-task continual-learning stream. Final, acquisition (Acq.), and mean stage success (Mean) are percentages; backward transfer (BWT) and forgetting (Forget) are percentage-point changes. Final success uses 250 episodes per method.

Method	Final ↑	Acq. ↑	Mean ↑	BWT ↑	Forget ↓
Matched source selection					
Random source	16.4	16.4	7.57	0.00	0.44
Relation-guided source	26.0	26.0	13.01	0.00	0.00
Other continual-learning methods					
CLARE	17.2	11.6	9.24	6.22	0.44
SequenFT	11.6	20.0	5.03	-9.33	15.56
ER	8.0	20.0	4.97	-13.33	14.67

We present the complete simulation stage curves and task illustrations alongside the real-robot results in Figure 7. The fixed simulation task order is: drag waterer, roll playing cards, rotate QR code, scan object, stamp seal, cut bread, push blocks to hit, beat block with hammer, press stapler, and erase board. We provide concise task descriptions and example settings below in Table 15, in learning order. Independent inheritance copies bottleneck MLP adapters across 18 layers. For each incoming task, the configuration uses 5,000 adapter updates, 2,000 reconstruction-autoencoder updates, and 1,000 batches for collecting routing statistics without gradient updates, with batch size 64. The random control samples a source skill uniformly from the previously learned tasks, fixes these selections for the run, and shares the initial task checkpoint. Optimizer and architecture settings are listed in Table 13 of Appendix D. Inference chooses adapters using layer-wise raw reconstruction errors.

The final evaluation uses 25 frozen, executable scenarios per task. Fixed-adapter ER trains one bottleneck adapter shared across the stream, with 18 layers of width 128 and 9,476,352 trainable parameters. It uses 5,000 policy updates per task, batch size 64, and 80% current-task versus 20% replay samples after the first task. ER shares the pretrained backbone and bottleneck architecture with the source-selection variants and uses the shared adapter without reconstruction routing. Its first-task checkpoint is trained separately from the checkpoint shared by random and relation-guided source selection. Original CLARE uses dynamic adapter sharing and expansion. Sequential FT also trains one fixed bottleneck adapter, with batch size 64 and no growing adapter library. Both fixed-adapter baselines use 5,000 updates per task and AdamW with constant learning rate 10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.999)$, $\epsilon = 10^{-8}$, and weight decay 10^{-10}. Optimizer states are reset at task boundaries; neither baseline uses EMA. Random and relation-guided source selection share one set of frozen evaluation manifests; fixed-adapter Sequential FT and ER share another. The two sets use the same task configurations and prompts, with 224 of 250 task–seed pairs in common.

E.2 EVALUATION METRICS

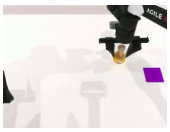
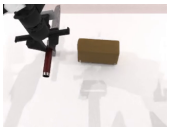
Let $P_{k,j}$ be the success rate on task j after learning stage k, with $j \le k$ and $K = 10$ tasks. The diagonal records acquisition success, and the final row records retained performance. We compute

$$\text{Final} = \frac{1}{K}\sum_{j=1}^{K} P_{K,j}, \qquad \text{Acq} = \frac{1}{K}\sum_{j=1}^{K} P_{j,j},$$

$$\text{Mean} = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{k}\sum_{j=1}^{k} P_{k,j}, \qquad \text{BWT} = \frac{1}{K-1}\sum_{j=1}^{K-1}(P_{K,j} - P_{j,j}). \tag{29}$$

$$\text{Forget} = \frac{1}{K-1}\sum_{j=1}^{K-1}\left(\max_{j\le k\le K} P_{k,j} - P_{K,j}\right).$$

Table 15: Simulation tasks in the continual-learning stream (stages 1–10).

Skill	Task Instruction	Task Setting
Drag	Grasp the waterer's handle and drag it to either side of the plant without lifting it, then release the handle.	
Roll	Pick up the playing cards with one arm, transfer them to the other arm, and flip them over.	
Rotate	Pick up the QR-code board from the table and rotate it so that the QR code faces the viewer.	
Scan	Pick up the scanner with one arm and the object with the other, then scan the object.	
Stamp	Grasp the stamp and press it onto the specified colored mat.	
Cut	Pick up the knife and use its blade to cut the bread into two halves.	
Push	Use horizontal fingertips to push block A across the table until it collides with the target block B.	
Hammer	Pick up the hammer and use it to strike the block on the table.	
Press	Use one arm to press the stapler.	
Erase	Pick up the eraser and wipe the colored area from the vertical board.	

Mean stage success averages the ten stage endpoints. BWT and forgetting average the first nine tasks, and the latter includes the final checkpoint when taking the maximum. We report success rates as percentages and changes as percentage points.

E.3 Real-Robot Evaluation

The real-robot task order is drag, fold, pick-place, shake, and wipe. We aggregate success counts from 10 trials in each of the 15 task–stage cells with $j \leq k$ per method, giving 50 final-stage trials per method. We compute the metrics in Eq. 29 with $K = 5$ over these cells. We illustrate the setup and objects with task images in Figure 7. We provide a short instruction and a setting image for each real-robot task below in Table 16, following the same task order.

Table 16: Real-robot tasks in the continual-learning stream, in learning order.

Skill	Task Instruction	Task Setting
Drag	Grasp the cart's strap and drag the cart across the table.	
Fold	Fold the towel on the table.	
Pick-place	Pick up the yellow object and place it in the mug.	
Shake	Grasp and shake the plastic bottle.	
Wipe	Use the towel to wipe the table.	