THOUGHTVLA: EVOLVING THOUGHT TRAJECTORIES FOR CONTINUAL VISION-LANGUAGE-ACTION ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Continual adaptation of pretrained vision-language-action (VLA) models aims to acquire new manipulation skills from sequentially arriving demonstrations while retaining performance on previously learned tasks. However, in progressive LoRA-MoE methods, updating the router can change the expert mixtures used for previously learned tasks even when their experts remain frozen. Meanwhile, pretrained semantic similarity alone may not reliably distinguish learned skills that are semantically related but require different manipulations. To address these challenges, we propose ThoughtVLA, a continual adaptation framework built around Continually Evolving Thought Trajectory (CETT), which formalizes how pre-action reasoning for skill selection evolves as task knowledge accumulates. In ThoughtVLA, Evolving Skill Reasoning selects a learned skill by interpreting the current input with pretrained semantics and accumulated task knowledge, and resolving ambiguity with task attributes. Each acquired skill retains a LoRA composition, which is used throughout execution when that skill is selected. Beyond skill reasoning, Complementary Knowledge Transfer supports new-skill acquisition by selecting prior LoRA updates that reduce the residual error left beyond the limited rank of new adapter and incorporating them into the new skill's LoRA composition. Across single-arm, bimanual, humanoid, mixed-embodiment, cross-backbone, and real-robot evaluations, ThoughtVLA achieves higher final success and lower forgetting than the compared methods.

1 INTRODUCTION

Vision-language-action (VLA) models such as RT-2 (Zitkovich et al., 2023), OpenVLA (Kim et al., 2025b), and π_0 (Black et al., 2025) learn visual and semantic representations together with control priors through pretraining. These priors support language-conditioned control across diverse objects, scenes, and manipulation tasks (Ghosh et al., 2024; Liu et al., 2025). However, deployed systems may encounter manipulation tasks whose requirements or operating conditions are poorly represented in the pretraining data. Adapting the pretrained policy with task-specific demonstrations builds on these priors to acquire skills for the new manipulation settings (Kim et al., 2025b;a).

In practice, demonstrations for new manipulation tasks may become available sequentially rather than as a single training set (Römer et al., 2026; Lei et al., 2026; Liang et al., 2026). Continual VLA adaptation (Liu et al., 2023; Wang et al., 2024a; Römer et al., 2026; Wang et al., 2026a) addresses this setting by acquiring skills for incoming manipulation tasks while retaining performance on previous ones. A common paradigm in continual VLA adaptation is progressive LoRA-MoE (Yu et al., 2025; Huai et al., 2025; Lei et al., 2026) , which expands a library of low-rank experts and uses an input-conditioned router for expert aggregation (Fig. 1a). However, this routing-based formulation remains vulnerable to routing drift, as later router updates can change the experts selected or their aggregation weights for previous-task inputs even when previous experts are frozen. Meanwhile, different instruction formulations of the same manipulation task can introduce ambiguity in task interpretation. (ichter et al., 2023; Driess et al., 2023; Huang et al., 2023; Zitkovich et al., 2023). These limitations motivate a thought-based alternative that interprets the task in relation to acquired skills, resolves ambiguity when needed, and identifies an appropriate learned skill before action.

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.

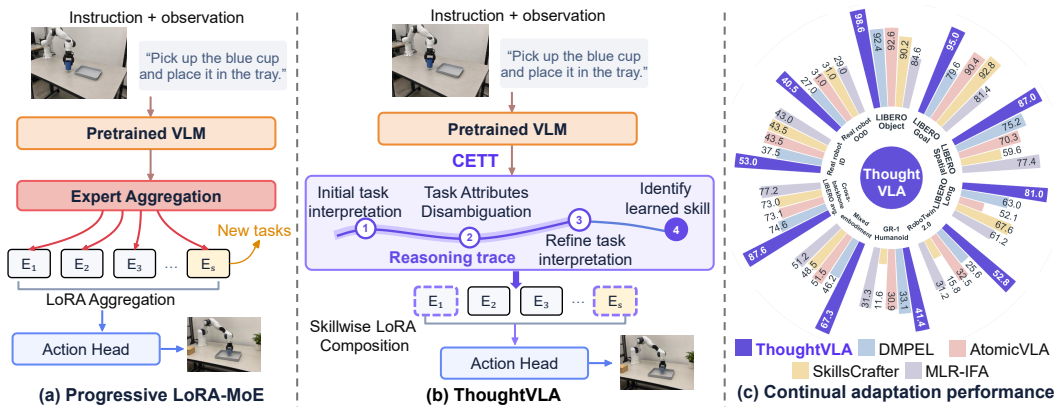

Figure 1: Skill acquisition and evolving task knowledge for reasoning. (a) Progressive LoRA-MoE uses pretrained features to aggregate expert outputs. (b) ThoughtVLA accumulates task knowledge with skills, using pretrained semantics to interpret manipulation tasks and select skills before action. CETT connects reasoning across stages. (c) Final success across settings. Better seen at magnification.

To address these challenges, we introduce a novel formulation of *Continually Evolving Thought Trajectory (CETT)*. Here, thought denotes a structured pre-action process that interprets the current manipulation task in relation to acquired skills, examines accumulated evidence when necessary, and refines ambiguous interpretations before identifying a learned skill for execution. A reasoning trace records the intermediate interpretations, consulted evidence, and resulting skill selection. CETT further captures how the knowledge underlying this process evolves across stages: newly acquired task knowledge expands the evidence available for reasoning, while accumulated statistics recalibrate semantic comparison and instruction matching over the growing skill library. CETT thus separates the execution policy associated with each acquired skill from the evolving knowledge used to identify it before action. Subsequent experience can change how the same input is interpreted and refined, while the pretrained encoder, previous skill policies, and reasoning rules remain unchanged.

Building on CETT, we propose ThoughtVLA, which replaces direct expert routing with thought-guided skill identification followed by skillwise LoRA composition. Evolving Skill Reasoning (ESR) uses accumulated task knowledge to semantically match the current observation and instruction to acquired skills. When candidate skills are difficult to distinguish, ESR refines the match using task attributes before selecting a skill for execution. The selected skill determines a LoRA composition, separating skill identification from the execution policy associated with that skill. Complementary Knowledge Transfer (CKT) supports new-skill acquisition by selecting previous frozen updates according to how much they reduce the error that remains after adaptation within the new adapter's rank budget. Before accepting a proposed transfer, CKT briefly fine-tunes a new adapter with and without the selected updates and validates the transfer on held-out demonstrations. Together, each acquired skill enters the growing library with its task knowledge and skillwise LoRA composition, allowing subsequent stages to reason over accumulated experience while expanding the system's manipulation capabilities.

Our contributions are:

- We formulate continual VLA adaptation in terms of skill acquisition and evolving task knowledge for pre-action reasoning. Within CETT, accumulated experience supports interpreting the current manipulation task in relation to acquired skills and refining ambiguous interpretations before action.
- We introduce ThoughtVLA, in which ESR uses accumulated task knowledge to relate the current manipulation task to prior experience and identify a learned skill before action, whereas CKT transfers knowledge from previous skills to support learning new skills.
- We evaluate ThoughtVLA on single-arm, bimanual, humanoid, and mixed-embodiment continual streams, with cross-backbone and real-robot in-domain and out-of-domain settings. ThoughtVLA achieves the highest final success and lowest forgetting among the compared methods across all evaluated settings.

2 Related Work

Vision-language-action models. VLA models (Li et al., 2024; Zhang et al., 2026a; Wang et al., 2026b) adapt pretrained multimodal representations to closed-loop robot control. RT-2 (Zitkovich et al., 2023) expresses actions as language tokens, OpenVLA (Kim et al., 2025b) provides an open pretrained policy, and π_0 (Black et al., 2025) generates continuous action chunks with flow matching. To support decisions before execution, other VLAs introduce deliberation alongside reactive control (Zawalski et al., 2025). Hume (Song et al., 2025) reasons over action candidates, DP-VLA (Han et al., 2024) separates high-level reasoning from execution, and PonderPounce (Choi et al., 2026) uses a pretrained multimodal model to interpret episode context. AtomicVLA (Zhang et al., 2026b) predicts atomic-skill sequences and routes them through an expandable expert library. Its continual evaluation adds one skill to a base model rather than following a multi-stage continual stream.

Continual Robot and VLA Adaptation. LIBERO (Liu et al., 2023) formalizes lifelong manipulation streams, while LOTUS (Wan et al., 2024) builds modular skill libraries. Continual VLA methods use latent replay (Yu et al., 2026), routed adapter expansion (Römer et al., 2026), or expert expansion with coefficient replay (Lei et al., 2026). Stellar VLA (Wu et al., 2026) combines evolving skill knowledge with mixture routing and replay, while LoRA with replay can exhibit little forgetting (Liu et al., 2026). Beyond robotics, ProgLoRA (Yu et al., 2025) expands expert mixtures, while RePro (Tran et al., 2026) routes isolated experts using frozen features and prototypes. O-LoRA (Wang et al., 2023), InfLoRA (Liang & Li, 2024), and KeepLoRA (Luo et al., 2026) organize updates through subspace or residual structure. Our focus is evolving task knowledge for skill reasoning alongside skill acquisition. Beyond a single feature-based assignment, ThoughtVLA uses accumulated task knowledge to revise ambiguous interpretations before selecting a learned skill. As knowledge accumulates, new experience can inform reasoning about previous skills. CKT selects knowledge from previous skills by its complementarity to new-skill learning rather than by a previous skill's performance on the new task.

3 Problem Setup and CETT Formulation

3.1 Continual VLA Adaptation

At stage $s \in \{1, \dots, S\}$, the learner receives demonstrations grouped by manipulation task, with known stage boundaries. Let $\mathcal{A}_s$ index the skills acquired at stage s, one per manipulation task, and $\mathcal{B}_q$ the demonstrations for skill q, giving $\mathcal{B}_s = \bigcup_{q \in \mathcal{A}_s} \mathcal{B}_q$. Each sample $(\mathbf{x}, \ell, \mathbf{u}) \in \mathcal{B}_q$ pairs observations $\mathbf{x}$ and an instruction ℓ with a target action $\mathbf{u}$. Training uses current-stage demonstrations without retaining previous raw trajectories. The skill library after stage s is $\mathcal{K}_s = \bigcup_{t=1}^{s} \mathcal{A}_t$. The VLA policy $F_{\boldsymbol{\theta}}$ predicts actions from $(\mathbf{x}, \ell)$ through a vision-language model (VLM) and an action head. Starting from StarVLA (Community, 2026), pretrained on Open X-Embodiment (Collaboration et al., 2025) data, ThoughtVLA freezes its base weights $\boldsymbol{\theta}_0$ and uses the VLM as semantic encoder φ_0 for images and instructions. At each stage, the action head learns one set of LoRA (Hu et al., 2021) updates $\boldsymbol{\omega}_s = \{\Delta \mathbf{W}_s^v\}_v$ from $\mathcal{B}_s$, where $\Delta \mathbf{W}_s^v = \mathbf{B}_s^v \mathbf{A}_s^v$ at adapted layer v. Skills in $\mathcal{A}_s$ share $\boldsymbol{\omega}_s$ but each retains its task knowledge and skillwise LoRA composition. After stage s, the retained parameters are $\boldsymbol{\theta}_s = (\boldsymbol{\theta}_0, \{\boldsymbol{\omega}_t\}_{t=1}^{s})$. Progressive LoRA-MoE methods (Lei et al., 2026) use an input-conditioned router to aggregate the growing set of expert outputs.

3.2 Continually Evolving Thought Trajectory

CETT formalizes thought-based reasoning over learned skills, linking acquired task knowledge to pre-action interpretation, refinement, and skill identification. Here, thought denotes this structured inference process rather than a generated natural-language chain of thought. After stage s, task knowledge $\mathcal{H}_s$ describes the manipulation requirements associated with skills in $\mathcal{K}_s$, supporting subsequent input interpretation. The skill-reasoning map G_s uses $\mathcal{H}_s$ and the pretrained encoder φ_0 to interpret the input and identify a skill in $\mathcal{K}_s$. Across the continual stream, we define CETT as $\mathfrak{C}_{1:S} = \big((\mathcal{H}_s, \mathcal{K}_s, G_s)\big)_{s=1}^{S}$.

Across stages, G_s uses updated task knowledge and the growing skill library with the same encoder and reasoning rules. Given $\boldsymbol{\chi} = (\mathbf{x}_0, \ell)$, where $\mathbf{x}_0$ denotes the initial images, G_s interprets the current

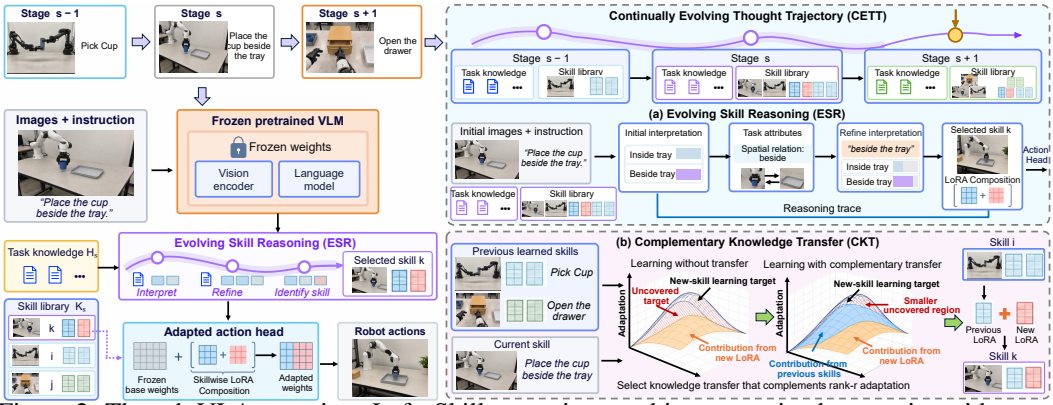

Figure 2: ThoughtVLA overview. Left: Skill reasoning combines pretrained semantics with accumulated task knowledge to identify a skill whose skillwise LoRA composition configures the action head. (a) ESR maintains CETT with task knowledge accumulated across stages, refining ambiguous interpretations to select a skill. (b) CKT evaluates previous-skill LoRA updates through low-rank complementarity and held-out validation before transfer.

manipulation task before action and returns a reasoning trace $\boldsymbol{\tau}$ together with a skill index $q \in \mathcal{K}_s$:

$$(\boldsymbol{\tau}, q) = G_s(\boldsymbol{\chi}), \qquad q \in \mathcal{K}_s. \tag{1}$$

The reasoning trace $\boldsymbol{\tau}$ records the interpretation, consulted evidence, and selected skill. That skill's execution map specifies its skillwise LoRA composition of weighted updates. CETT therefore links skill acquisition with evolving task knowledge: experience gained from learning new skills informs how later manipulation tasks are interpreted and related to the growing skill library.

4 Method

Overview. ThoughtVLA (Fig. 2) realizes CETT through pre-action skill reasoning and skillwise LoRA composition. At each adaptation stage, ESR (Sec. 4.1) incorporates information from the incoming demonstrations into the accumulated task knowledge used for skill selection. CKT (Sec. 4.2) supports new-skill learning by selecting previous-skill LoRA updates that complement the new low-rank adaptation and validating proposed transfers on held-out demonstrations. The resulting skill and updated task knowledge become available at stage completion. At inference, ESR uses pretrained multimodal representations of the initial images and instruction to match the current task to acquired skills. When the match is ambiguous, it refines the interpretation using accumulated instruction evidence and task attributes before selecting a skill. A reasoning trace records this process, and the selected skill's LoRA composition configures the action head throughout the episode.

4.1 Evolving Skill Reasoning

ESR realizes CETT's pre-action thought process by relating the initial images and instruction to prior experience. It forms an initial task interpretation and, when ambiguity remains, consults further evidence to revise it before identifying a learned skill. The intermediate scores, consulted evidence, and selected skill form the reasoning trace of this process.

Forming and examining an interpretation requires both a semantic connection to learned skills and information about their manipulation requirements. For each skill q, ESR therefore retains $\boldsymbol{\eta}_q$, comprising a semantic representation, information from demonstration instructions, and task attributes. The semantic representation $\boldsymbol{\mu}_q$ is the mean of normalized VLM features from the record-construction subset of $\mathcal{B}_q$. To compare the current input and learned-skill representations under a common normalization, ESR also uses statistics $\mathbf{S}_s$ accumulated across learned skills. These two components constitute the task knowledge introduced in Sec. 3: $\mathcal{H}_s = \big(\{\boldsymbol{\eta}_q\}_{q\in\mathcal{K}_s}, \mathbf{S}_s\big)$, where $\mathbf{S}_s = \sum_{q\in\mathcal{K}_s} \boldsymbol{\xi}_q$. The statistics $\boldsymbol{\xi}_q$ in $\boldsymbol{\eta}_q$ comprise sample counts, feature moments, and document frequencies that determine instruction-matching weights.

Comparing the VLM representation of the current images and instruction with each retained skill representation yields the initial semantic scores:

$$[\mathbf{c}_s^0]_q = \mathsf{N}_s(\varphi_0(\boldsymbol{\chi}))^\top \mathsf{N}_s(\boldsymbol{\mu}_q). \tag{2}$$

Here, $\mathbf{c}_s^0 \in \mathbb{R}^{|\mathcal{K}_s|}$ contains one semantic score per learned skill. The operator N_s applies shared semantic normalization using the accumulated feature statistics in $\mathbf{S}_s$. To refine this initial interpretation, ESR considers up to K_c highest-scoring skills, forming the candidate set $\mathcal{C}_s = \mathrm{TopK}_{K_c}(\mathbf{c}_s^0)$. When leading semantic scores are close, instruction matching adjusts them within $\mathcal{C}_s$ only if it distinguishes the candidates, yielding $\mathbf{c}_s^1$. Otherwise, $\mathbf{c}_s^1 = \mathbf{c}_s^0$; matching details are in Appendix B.

An initial interpretation can relate the current input to familiar skills while leaving specific manipulation requirements unresolved. To examine these requirements, ESR considers task attributes: spatial relation, manner, direction, and count. When ambiguity persists, it compares the available input attributes with those associated with candidate skills. For example, an interpretation favoring placement inside a tray can be revised when the instruction specifies placement beside it. Attribute agreement thus provides a basis for revising the interpretation before committing to a skill:

$$[\mathbf{c}_s^2]_q = [\mathbf{c}_s^1]_q + g_s^{\mathrm{attr}} \lambda_{\mathrm{attr}} [\bar{\mathbf{e}}_s^{\mathrm{attr}}]_q. \tag{3}$$

Here, $\bar{\mathbf{e}}_s^{\mathrm{attr}}$ contains attribute-agreement scores standardized within $\mathcal{C}_s$, and the stage-independent weight λ_{attr} controls their contribution. The gate g_s^{attr} is shared by the candidate set: it is one when initial score correction is triggered, ambiguity persists, and available attributes distinguish the candidates; otherwise, it is zero. Zero attribute gating preserves $\mathbf{c}_s^1$; scores outside $\mathcal{C}_s$ remain at their $\mathbf{c}_s^0$ values. The interpretation can therefore be revised using specific requirements rather than being determined solely by the initial semantic preference.

After refinement, ESR selects $q = \arg\max_{k \in \mathcal{K}_s} [\mathbf{c}_s^2]_k$, as returned in Eq. (1). The reasoning trace retains the intermediate scores and consulted evidence leading to this choice:

$$\boldsymbol{\tau} = \left(\mathbf{c}_s^0, \mathbf{e}_s^{(1)}, \mathbf{c}_s^1, \mathbf{e}_s^{(2)}, \mathbf{c}_s^2, q\right). \tag{4}$$

The entries $\mathbf{e}_s^{(1)}$ and $\mathbf{e}_s^{(2)}$ record evidence for initial score correction and task-attribute refinement, respectively. A skipped correction or refinement leaves its evidence entry empty and the scores unchanged. The execution map $\mathcal{I}_s(q)$ specifies the skillwise LoRA composition for the selected skill q; applying it yields the action-head weights:

$$\mathcal{I}_s(q) = \left\{(\Delta\mathbf{W}_t^v, \beta_{q,t}^v)\right\}_{t \le s_q, v}, \qquad \mathbf{W}_q^v = \mathbf{W}_0^v + \sum_{t=1}^{s_q} \beta_{q,t}^v \Delta\mathbf{W}_t^v. \tag{5}$$

Here, s_q is the stage at which skill q is acquired, $\mathbf{W}_0^v$ is the pretrained weight, and $\beta_{q,t}^v$ is skill q's coefficient for the stage-t LoRA update at layer v. The map contains only updates with nonzero coefficients. Using the skillwise LoRA composition throughout the episode, the VLA generates actions $\mathbf{u}_h = F_{\boldsymbol{\theta}_s, \mathcal{I}_s}(\mathbf{x}_h, \ell, q)$ from observations $\mathbf{x}_h$ at step h. At stage completion, ESR incorporates information from each new skill $q \in \mathcal{A}_s$ into $\mathcal{H}_{s-1}$ to form $\mathcal{H}_s$ for subsequent reasoning:

$$\mathbf{S}_s = \mathbf{S}_{s-1} + \sum_{q \in \mathcal{A}_s} \boldsymbol{\xi}_q, \quad \mathcal{H}_s = \left(\{\boldsymbol{\eta}_q\}_{q \in \mathcal{K}_{s-1}} \cup \{\boldsymbol{\eta}_q\}_{q \in \mathcal{A}_s}, \mathbf{S}_s\right). \tag{6}$$

New skills expand the evidence for subsequent inputs, while updated $\mathbf{S}_s$ recalibrates semantic normalization and instruction matching for new and previous skills. The same input can thus yield different scores, refinements, and reasoning traces as experience accumulates. CETT captures this evolution in the knowledge underlying pre-action thought, while previous skills and policies, the encoder, and reasoning rules remain unchanged.

4.2 Complementary Knowledge Transfer

For each incoming skill q, CKT decides whether and which previous skill to transfer from relative to what the new rank-r stage update can learn. A previous skill may provide complementary updates even when it cannot complete the new manipulation task on its own. The following local response approximation makes this comparison explicit. For $q \in \mathcal{A}_s$, CKT considers no transfer ($\varnothing$) or

a previous skill $k \in \mathcal{K}_{s-1}$. The candidate $\mathcal{F} = \mathcal{I}_{s-1}^{\mathrm{CKT}}(k)$ contains all of skill k's updates and coefficients, including previous transfers.

At frozen base activations, CKT compares each candidate's response with the action-loss gradient target for q. For $\mathcal{F} = \mathcal{I}_{s-1}^{\mathrm{CKT}}(k)$, its combined update is $\mathbf{D}_{\mathcal{F}}^{v} = \mathbf{W}_{k}^{v} - \mathbf{W}_{0}^{v}$ from Eq. (5); $\mathbf{D}_{\varnothing}^{v} = \mathbf{0}$. Subtracting the candidate response leaves the residual for the new update:

$$\mathbf{Y}_{q}^{v} = -\gamma_{\mathrm{geo}}\big(\nabla_{\mathbf{W}^{v}}\mathcal{E}_{q}^{\mathrm{write}}\big)\overline{\mathbf{H}}_{q}^{v}/c_{v}, \qquad \mathbf{R}_{q,\mathcal{F}}^{v} = \mathbf{Y}_{q}^{v} - \mathbf{D}_{\mathcal{F}}^{v}\overline{\mathbf{H}}_{q}^{v}/c_{v}. \tag{7}$$

At frozen base weights $\boldsymbol{\theta}_0$, $\mathcal{E}_{q}^{\mathrm{write}}$ is mean action loss on $\mathcal{B}_{q}^{\mathrm{write}}$; $\overline{\mathbf{H}}_{q}^{v}$ contains the compressed layer-v inputs. The constants $c_v > 0$ and γ_{geo} normalize responses and scale the gradient step. Thus $\mathbf{R}_{q,\mathcal{F}}^{v}$ is the response left for the new update after candidate transfer.

Whether transfer is complementary depends on the structure of this remaining response, not only its magnitude. With activations held constant, a rank-r parameter update produces a response of rank at most r. CKT therefore evaluates the error remaining after the best rank-r approximation to the residual, rather than penalizing the part that this approximation can represent. The Eckart–Young–Mirsky theorem (Wang et al., 2024b) yields ϵ_r, which CKT uses to define Δ_r for skill q:

$$\epsilon_r(\mathbf{A}) = \min_{\mathrm{rank}(\mathbf{Z})\le r} \|\mathbf{A}-\mathbf{Z}\|_F^2 = \sum_{a>r}\sigma_a^2(\mathbf{A}), \quad \Delta_r(q,\mathcal{F}) = \sum_{v} w_v\left[\epsilon_r(\mathbf{Y}_q^v) - \epsilon_r(\mathbf{R}_{q,\mathcal{F}}^v)\right]. \tag{8}$$

Here, $\sigma_a(\mathbf{A})$ is the a-th largest singular value, and w_v are nonnegative layer weights satisfying $\sum_v w_v = 1$. With activations held constant and only a rank-at-most-r constraint, the new update can attain this approximation (Appendix A). Because $\Delta_r(q,\varnothing) = 0$, a positive score indicates that transfer leaves less error beyond what the new rank-r update can represent than no transfer. CKT proposes a previous skill with the highest score only when that score is positive. This local criterion measures complementarity with transfer fully active, as at inference. CKT then tests the proposal through paired adaptation trials on $\mathcal{B}_{q}^{\mathrm{fit}}$, with and without transfer. On $\mathcal{B}_{q}^{\mathrm{val}}$, CKT accepts $\mathcal{F}_{q}^{\star}$ if its mean trajectory-wise relative action-loss reduction over the no-transfer trial meets the acceptance threshold and more than half the trajectories improve. If the highest score is nonpositive or confirmation fails, CKT sets $\mathcal{F}_{q}^{\star} = \varnothing$ without testing another candidate. Candidate scoring, trial matching, and acceptance details are in Appendix B. CKT then discards the trial updates and trains one rank-r stage update $\boldsymbol{\omega}_s$ on $\mathcal{B}_{s}^{\mathrm{train}}$ with previous updates frozen. Each $q \in \mathcal{A}_s$ combines its accepted previous updates with that shared update: $\mathcal{I}_{s}^{\mathrm{CKT}}(q) = \mathcal{F}_{q}^{\star} \cup \{(\Delta\mathbf{W}_{s}^{v}, 1)\}_{v}$, which specifies the action-head weights in Eq. (5). At stage completion, the new skills and their task knowledge enter $\mathcal{K}_s$ and $\mathcal{H}_s$. When ESR selects q, the VLA generates actions with $\mathcal{I}_{s}^{\mathrm{CKT}}(q)$, combining the new LoRA updates and any accepted transfers.

Stage-wise adaptation. At stage s, all incoming skills are learned from the same pre-stage state, with CKT transferring only from previously acquired skills in $\mathcal{K}_{s-1}$. After adaptation, ThoughtVLA retains the new LoRA update and the resulting skillwise LoRA compositions, while ESR updates the accumulated task knowledge. The expanded library then supports subsequent reasoning and learning.

Inference. Given the initial observations and instruction, ESR produces a reasoning trace and identifies a skill q. Its stored skillwise LoRA composition $\mathcal{I}_{s}^{\mathrm{CKT}}(q)$ configures the action head throughout the episode, without rerunning CKT. Task knowledge, LoRA updates, and composition records accumulate as the skill library grows.

5 Experiments

5.1 Experimental Setup

Continual learning protocol. Table 2 summarizes the disjoint groups of manipulation tasks introduced at the initial and subsequent stages. All methods receive the same current-stage demonstrations; ThoughtVLA retains no previous raw trajectories, while replay baselines use their stated budgets. For single-arm manipulation, we use the Object, Goal, Spatial, and Long suites of LIBERO (Liu et al., 2023). The other single-embodiment streams cover bimanual RoboTwin 2.0 (Chen et al., 2025) and GR-1 Humanoid tasks from RoboCasa GR-1 Tabletop (Nasiriany et al., 2024; NVIDIA et al., 2025), with a higher-dimensional action interface. The mixed stream presents 30 LIBERO, 15 RoboTwin, and 10 GR-1 manipulation tasks. For acquired RoboTwin skills, evaluation uses unseen instruction formulations that are excluded from adaptation data.

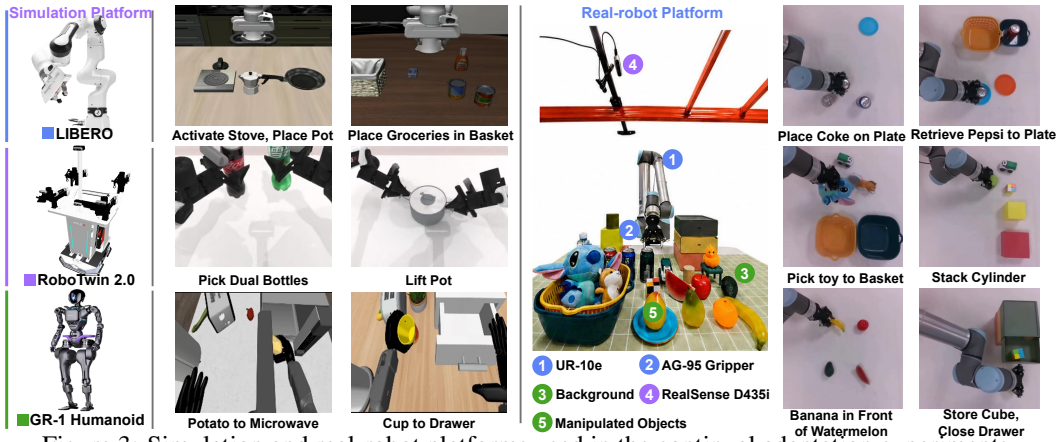

Figure 3: Simulation and real-robot platforms used in the continual adaptation experiments.

Table 1: **Single-arm continual adaptation.** Closed-loop results on four LIBERO streams. Purple bold and blue underlining mark the best and second-best listed values.

Method	Object			Goal			Spatial			Long		
	FSR ↑	CSR ↑	Forg. ↓	FSR ↑	CSR ↑	Forg. ↓	FSR ↑	CSR ↑	Forg. ↓	FSR ↑	CSR ↑	Forg. ↓
Sequential FT	31.4	44.8	73.6	9.8	26.0	94.7	10.0	24.4	88.2	10.0	23.8	83.3
O-LoRA (EMNLP Findings'23)	44.6	34.0	21.3	54.8	38.8	12.7	67.6	36.5	6.9	37.0	26.3	45.6
CL-MoE (CVPR'25)	60.4	70.3	20.3	70.2	74.5	18.2	70.6	70.0	22.3	50.2	62.4	32.9
ProgLoRA (ACL Findings'25)	51.2	54.2	32.4	43.8	56.6	36.5	72.4	80.2	15.4	10.2	43.2	68.4
SkillsCrafter (AAAI'26)	90.2	89.7	9.3	92.8	93.0	4.9	59.6	46.8	15.6	67.6	70.9	15.7
DMPEL (TMLR'26)	92.4	93.8	8.2	79.6	85.0	11.9	75.2	80.2	12.6	63.0	72.3	17.6
AtomicVLA (CVPR'26)	92.6	92.1	6.3	90.4	92.6	5.0	70.4	78.4	8.9	52.2	64.7	20.8
MLR-IFA (CVPR'26)	84.6	91.7	16.9	81.4	68.0	16.9	77.4	78.0	14.9	61.2	70.1	31.6
CLARE (RA-L'26)	68.8	75.5	18.4	77.8	83.2	14.6	76.4	70.2	16.8	58.2	64.3	25.8
ThoughtVLA	**98.6**	**99.5**	**0.6**	**95.0**	**95.5**	**1.2**	**87.0**	**86.8**	**0.5**	**81.0**	**83.4**	**1.4**

Implementation details. The primary policy pairs Qwen3-VL-4B (Bai et al., 2025) with a GR00T-style DiT action head (NVIDIA et al., 2025) trained using flow matching (Lipman et al., 2023). Cross-backbone evaluation repeats the LIBERO streams with Qwen3.5-2B and an OpenVLA-OFT-style continuous-action head (Kim et al., 2025a). ThoughtVLA freezes pretrained weights and trains rank-16 action-head LoRA updates. Within each setting, all methods start from the same Open X-Embodiment-pretrained StarVLA VLA checkpoint (Community, 2026) (Appendix B). After each stage, we evaluate all encountered manipulation tasks without oracle skill or stage IDs. Let $a_{s,q}$ be the success rate on skill q's manipulation task after stage s, and s_q its learning stage. Final average success rate is FSR $= |\mathcal{K}_S|^{-1} \sum_{q \in \mathcal{K}_S} a_{S,q}$; current-stage average success rate is CSR $= S^{-1} \sum_{s=1}^{S} |\mathcal{K}_s|^{-1} \sum_{q \in \mathcal{K}_s} a_{s,q}$. Forgetting averages the nonnegative decline from post-learning success over subsequent stages for each previous skill, then across those skills: Forg. $= |\mathcal{K}_{S-1}|^{-1} \sum_{q \in \mathcal{K}_{S-1}} (S - s_q)^{-1} \sum_{t=s_q+1}^{S} \max\{0, a_{s_q,q} - a_{t,q}\}$. Further implementation details are provided in the supplementary material.

Table 2: **Continual evaluation settings.** Task counts refer to manipulation tasks. The split $a+b\times c$ denotes a initial tasks and c later stages of b each.

Setting	Embodiment	Continual setting	
		Tasks	Stage split
LIBERO (four suites)	Single-arm	10 each	$5+1\times5$
RoboTwin 2.0	Bimanual	50	$25+5\times5$
GR-1 Humanoid	Humanoid	24	$12+2\times6$
Mixed-embodiment	Heterogeneous	55	$5+1\times50$
Cross-backbone	Single-arm	10 each	$5+1\times5$
Real-robot (UR-10e)	Single-arm	20	$10+2\times5$

5.2 Continual Adaptation in Simulation

Across the simulation settings (Tabs. 1 and 3), ThoughtVLA reaches 90.4% mean LIBERO FSR, exceeding the best baseline by 2.2–13.4 points across streams. Its gains on RoboTwin 2.0, GR-1, and the mixed stream are 20.3, 8.3, and 15.8 points, respectively. Forgetting remains at 0.5–1.4 points on LIBERO, 2.0 on GR-1, 2.1 on RoboTwin 2.0, and 3.4 on the mixed stream. High final

Table 3: **Continual adaptation across embodiments.** Grouped columns report final, not stage-wise, success: (a) initial and later tasks; (b) single-arm, bimanual, and humanoid groups in learning order.

(a) Single embodiment	RoboTwin 2.0					GR-1				
Method	1–25	26–50	FSR ↑	CSR ↑	Forg. ↓	1–12	13–24	FSR ↑	CSR ↑	Forg. ↓
Sequential FT	4.7	10.2	7.4	12.1	43.9	24.2	25.2	24.7	28.6	18.3
O-LoRA (EMNLP Findings'23)	17.4	38.5	28.0	30.6	26.2	31.8	29.2	30.5	32.9	10.4
CL-MoE (CVPR'25)	15.4	40.6	28.0	35.4	25.7	28.5	29.7	29.1	31.2	11.5
ProgLoRA (ACL Findings'25)	30.6	32.1	31.4	42.2	7.9	20.7	19.0	19.8	23.2	10.6
SkillsCrafter (AAAI'26)	20.4	11.2	15.8	20.2	5.7	10.3	12.9	11.6	12.2	5.6
DMPEL (TMLR'26)	20.4	30.8	25.6	30.0	20.3	32.4	33.7	33.1	36.2	6.9
AtomicVLA (CVPR'26)	30.2	34.8	32.5	40.0	6.4	31.7	30.2	30.9	32.8	10.0
MLR-IFA (CVPR'26)	16.1	46.4	31.2	39.8	22.2	31.0	31.5	31.3	35.0	8.6
CLARE (RA-L'26)	20.2	32.7	26.4	33.2	28.7	29.9	32.1	31.0	33.2	9.1
ThoughtVLA	**58.8**	**46.8**	**52.8**	**57.4**	**2.1**	**45.0**	**37.8**	**41.4**	**43.6**	**2.0**

(b) Mixed embodiment	Final success by task group			All 55 tasks		
Method	Single-arm 1–30	Bimanual 31–45	Humanoid 46–55	FSR ↑	CSR ↑	Forg. ↓
Sequential FT	0.4	10.4	23.7	7.4	16.2	78.9
O-LoRA (EMNLP Findings'23)	45.9	21.7	29.3	36.3	45.2	38.5
CL-MoE (CVPR'25)	60.9	32.8	30.7	47.8	54.2	31.0
ProgLoRA (ACL Findings'25)	49.1	31.3	27.8	40.4	44.8	5.8
SkillsCrafter (AAAI'26)	73.4	23.5	11.3	48.5	53.7	6.0
DMPEL (TMLR'26)	62.8	30.0	20.4	46.2	50.6	12.3
AtomicVLA (CVPR'26)	70.1	28.7	30.2	51.5	56.0	5.7
MLR-IFA (CVPR'26)	67.3	33.5	29.4	51.2	58.3	23.1
CLARE (RA-L'26)	64.7	32.2	28.9	49.3	54.3	26.7
ThoughtVLA	**84.6**	**53.4**	**36.2**	**67.3**	**71.5**	**3.4**

Table 4: **Real-robot continual adaptation.** Each condition uses 10 rollouts per manipulation task per stage. Index ranges give final success for initial and later groups, not stage boundaries.

Method	In-domain					Out-of-domain				
	1–10	11–20	FSR ↑	CSR ↑	Forg. ↓	1–10	11–20	FSR ↑	CSR ↑	Forg. ↓
ProgLoRA (ACL Findings'25)	25.0	24.0	24.5	30.4	6.7	22.0	18.0	20.0	21.6	6.1
SkillsCrafter (AAAI'26)	47.0	40.0	43.5	49.3	5.6	34.0	28.0	31.0	34.5	6.1
DMPEL (TMLR'26)	42.0	33.0	37.5	41.8	7.2	30.0	24.0	27.0	31.6	6.7
AtomicVLA (CVPR'26)	45.0	42.0	43.5	46.8	5.0	35.0	27.0	31.0	34.0	6.1
MLR-IFA (CVPR'26)	49.0	37.0	43.0	50.4	11.1	32.0	26.0	29.0	32.4	10.0
CLARE (RA-L'26)	33.0	32.0	32.5	42.7	18.3	28.0	22.0	25.0	28.6	16.1
ThoughtVLA	**56.0**	**50.0**	**53.0**	**55.1**	**2.2**	**50.0**	**31.0**	**40.5**	**42.5**	**3.3**

success and low forgetting indicate sustained task performance as the library grows, contrasting with routing-drift-induced forgetting in progressive LoRA-MoE baselines with frozen previous experts.

On RoboTwin 2.0, ThoughtVLA reaches 52.8% FSR under unseen instructions, supporting ESR's aim to relate unfamiliar formulations to acquired skills before their skillwise LoRA compositions generate actions. In the mixed stream, the initial single-arm skills retain 84.6% final success after bimanual and humanoid adaptation, while CSR reaches 71.5%. Together, these results indicate that ThoughtVLA continues to apply acquired skills as its library expands across embodiments.

Cross-backbone and efficiency. ThoughtVLA leads FSR and CSR across four LIBERO streams with Qwen3.5-2B and an OpenVLA-OFT-style head. On LIBERO-Object and RoboTwin 2.0, it has the least parameter growth and persistent storage and second-lowest training time (Appendix C.1).

5.3 Real-Robot Continual Adaptation

We evaluate ThoughtVLA on a UR-10e with an AG-95 gripper and a stationary third-person RealSense D435i camera across 20 manipulation tasks, each with 50 demonstrations. These tasks span five categories: goal-directed manipulation, receptacle interaction, spatial-relation reasoning, structured manipulation, and multi-step manipulation. After each stage, we evaluate every encountered manipulation task using 10 in-domain (ID) and 10 out-of-domain (OOD) rollouts per task. We report

ID performance under the demonstration conditions and OOD performance under changes to object position, goal position, clutter, lighting, or background, varying one factor at a time. In real-robot evaluation (Tab. 4), ThoughtVLA reaches 53.0% ID and 40.5% OOD FSR, both 9.5 percentage points above the best baselines. ID CSR reaches 55.1%, 4.7 points above the best baseline; ID and OOD forgetting are 2.2 and 3.3 points, respectively. These results support using acquired skills under ID and OOD conditions, consistent with relating varied inputs to accumulated experience.

5.4 Ablation Studies

Table 5: Overall component ablation.

Variant	SR	CR	CKT	LIBERO-Goal FSR	LIBERO-Goal CSR	LIBERO-Goal Forg.	LIBERO-Long FSR	LIBERO-Long CSR	LIBERO-Long Forg.
Base	×	×	×	43.8	56.6	36.5	10.2	43.2	68.4
Base + SR (static)	✓	×	×	75.4	78.9	18.9	47.6	50.5	22.4
Base + SR	✓	×	×	80.2	83.8	12.5	52.8	58.1	18.7
Base + ESR	✓	✓	×	93.0	93.5	1.3	70.2	70.4	**1.4**
ThoughtVLA	✓	✓	✓	**95.0**	**95.5**	**1.2**	**81.0**	**83.4**	**1.4**

Overall component ablation. Table 5 compares skill reasoning (SR), conditional refinement (CR), and CKT on LIBERO-Goal and Long without oracle IDs. Base uses ProgLoRA (Yu et al., 2025); Base + SR tests the shift from routing to thought-based skill reasoning, raising FSR to 80.2% and 52.8%. Base + SR (static) fixes shared calibration at stage 1 but retains the same LoRA updates, skills, and reasoning rules; FSR falls to 75.4% and 47.6%. Shared recalibration adds 4.8 and 5.2 FSR points and reduces forgetting by 6.4 and 3.7 points, supporting its role in skill reasoning. CR raises FSR to 93.0% and 70.2%, with 1.3 and 1.4 points of forgetting, supporting conditional refinement before skill identification. With ESR, CKT further improves FSR on both streams.

Table 6: Skill reasoning and retention analysis.

Selector	SR acc. ↑	FSR ↑	CSR ↑	Forg. ↓
Semantic ($\mathbf{c}_s^0$)	58.2	28.5	30.6	31.9
ESR w/o attributes ($\mathbf{c}_s^1$)	82.3	44.2	50.7	4.0
ESR w/o gating	80.5	41.3	48.2	6.8
ESR ($\mathbf{c}_s^2$)	**91.0**	**52.8**	**57.4**	**2.1**
Oracle skill	100.0	55.0	58.8	0.2

ESR: conditional reasoning. Table 6 compares selectors on RoboTwin 2.0 using the same CKT-trained policies, task knowledge, and semantic normalization at each stage. Semantic already uses the calibrated scores $\mathbf{c}_s^0$, so the comparison evaluates refinement beyond semantic matching. Skill reasoning accuracy is $\text{SR acc.} = 100\ \text{Avg}_i\ \mathbf{1}[\hat{q}_i = q_i]$, where $\hat{q}_i$ and q_i are the selected and target skills at stage s_i. Attribute refinement adds 8.7 SR acc. and 8.6 FSR points over ESR w/o attributes. Removing the correction and ambiguity prerequisites lowers SR acc. to 80.5% and raises forgetting to 6.8 points, versus 91.0% and 2.1 for ESR, supporting conditional refinement before action. Oracle skill invokes the composition associated with the intended manipulation task under the same evaluation conditions. Its FSR, CSR, and forgetting provide a reference for execution and retention without skill-identification errors.

Table 7: CKT ablations on LIBERO-Long.

Variant	FSR ↑	CSR ↑	Forg. ↓
No transfer	70.2	70.4	**1.4**
Semantic similarity	77.6	78.4	1.6
Gradient alignment	75.8	75.5	1.6
Full residual	79.4	81.2	1.7
CKT (Δ_r)	**81.0**	**83.4**	**1.4**

CKT: complementary knowledge transfer. Table 7 compares transfer strategies on LIBERO-Long with ESR and matched new-update ranks and final-training budgets. No transfer learns new LoRA updates alone; semantic similarity ranks previous skills by demonstration VLM features, and gradient alignment compares their updates with the negative action-loss gradient. Full residual minimizes total error; CKT scores error beyond the new rank-r update's capacity. Transfer variants share the candidate pool, paired trials, and held-out acceptance; CKT requires a positive maximum Δ_r. Relative to no transfer, semantic similarity, and gradient alignment, CKT gains 10.8, 3.4, and 5.2 FSR points, respectively, with no additional forgetting. Its 1.6 FSR and 2.2 CSR point gains over full residual support low-rank complementarity.

6 Conclusion

ThoughtVLA grounds pre-action thought in evolving task knowledge during skill acquisition. CETT relates experience to task interpretation and skill identification. ESR invokes the identified skill's skillwise LoRA composition; CKT transfers previous LoRA updates for new-skill learning. Across two VLA backbones, multiple embodiments, mixed-embodiment streams, and real-robot evaluations, ThoughtVLA achieves higher final success and lower forgetting than the compared methods without retaining previous raw trajectories. This supports knowledge-guided skill acquisition and reasoning.

AI USE STATEMENT

We used generative AI tools to help organize literature, experimental results, and tables; revise manuscript text; and provide suggestions on method formulation, experimental design, and result interpretation. The authors evaluated AI-assisted suggestions and take responsibility for the final methods, experiments, results, and manuscript.

ETHICS STATEMENT

This work evaluates robotic manipulation in simulation and on a physical robot. During physical evaluation, predefined operating limits constrain the robot to a configured safety range within the experimental workspace. Sources and licenses for externally obtained simulation data are identified in the supplementary material.

REPRODUCIBILITY STATEMENT

Sections 3–5 specify the continual-learning protocol, method, evaluation metrics, and principal comparisons. The supplementary material provides the assumptions and analysis underlying CKT, implementation and fair-comparison details, task streams and evaluation protocols, and additional results across task orders and training seeds. These materials document the settings used to obtain and assess the reported results.

REFERENCES

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A Vision-Language-Action Flow Model for General Robot Control. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025. doi: 10.15607/RSS.2025.XXI.010.

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.

Suhwan Choi, Jaeyoon Jung, Sungkyung Kim, Yunsung Lee, and Youngjae Yu. Ponderpounce: A pretrained mllm as an episode context engine for robot control. *arXiv preprint arXiv:2608.24115*, 2026.

Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhinav Gupta, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2025.

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

Danny Driess, Fei Xia, Mehdi S. M. Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. PaLM-e: An embodied multimodal language model. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 8469–8488. PMLR, 23–29 Jul 2023.

Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, et al. Octo: An Open-Source Generalist Robot Policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.090.

ByungOk Han, Jaehong Kim, and Jinhyeok Jang. A dual process vla: Efficient robotic manipulation leveraging vlm. *arXiv preprint arXiv:2410.15549*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Tianyu Huai, Jie Zhou, Xingjiao Wu, Qin Chen, Qingchun Bai, Ze Zhou, and Liang He. Cl-moe: Enhancing multimodal large language model with dual momentum mixture-of-experts for continual visual question answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19608–19617, June 2025.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. Inner monologue: Embodied reasoning through planning with language models. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 1769–1782. PMLR, 14–18 Dec 2023.

brian ichter, Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, et al. Do as i can, not as i say: Grounding language in robotic affordances. In Karen Liu, Dana Kulic, and Jeff Ichnowski (eds.), *Proceedings of The 6th Conference on Robot Learning*, volume 205 of *Proceedings of Machine Learning Research*, pp. 287–318. PMLR, 14–18 Dec 2023.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-Tuning Vision-Language-Action Models: Optimizing Speed and Success. In *Proceedings of Robotics: Science and Systems*, LosAngeles, CA, USA, June 2025a. doi: 10.15607/RSS.2025.XXI.017.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan P Foster, Pannag R Sanketi, Quan Vuong, et al. Openvla: An open-source vision-language-action model. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 2679–2713. PMLR, 06–09 Nov 2025b.

Yuheng Lei, Sitong Mao, Shunbo Zhou, Hongyuan Zhang, Xuelong Li, and Ping Luo. Dynamic mixture of progressive parameter-efficient expert library for lifelong robot learning. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856.

Xinghang Li, Minghuan Liu, Hanbo Zhang, Cunjun Yu, Jie Xu, Hongtao Wu, Chilam Cheang, Ya Jing, Weinan Zhang, Huaping Liu, et al. Vision-language foundation models as effective robot imitators. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 26703–26721, 2024.

Wenqi Liang, Gan Sun, Yao He, Yu Ren, Jiahua Dong, and Yang Cong. Never-ending behavior-cloning agent for robotic manipulation. *IEEE Transactions on Neural Networks and Learning Systems*, 2026.

Yan-Shuo Liang and Wu-Jun Li. Inflora: Interference-free low-rank adaptation for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23638–23647, June 2024.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2023.

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

Huihan Liu, Changyeon Kim, Bo Liu, Minghuan Liu, and Yuke Zhu. Pretrained vision-language-action models are surprisingly resistant to forgetting in continual learning. *arXiv preprint arXiv:2603.03818*, 2026.

Songming Liu, Lingxuan Wu, Bangguo Li, Hengkai Tan, Huayu Chen, Zhengyi Wang, Ke Xu, Hang Su, and Jun Zhu. Rdt-1b: a diffusion foundation model for bimanual manipulation. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 29982–30009, 2025.

Mao-Lin Luo, Zi-Hao Zhou, Yi-Lin Zhang, Yuanyu Wan, Min-Ling Zhang, and Tong Wei. Keeplora: Continual learning with residual gradient adaptation. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 136017–136040, 2026.

Soroush Nasiriany, Abhiram Maddukuri, Lance Zhang, Adeet Parikh, Aaron Lo, Abhishek Joshi, Ajay Mandlekar, and Yuke Zhu. RoboCasa: Large-Scale Simulation of Household Tasks for Generalist Robots. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, July 2024. doi: 10.15607/RSS.2024.XX.050.

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

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*, 11(7):8331–8338, 2026. ISSN 2377-3774. doi: 10.1109/lra.2026.3693992.

Haoming Song, Delin Qu, Yuanqi Yao, Qizhi Chen, Qi Lv, Yiwen Tang, Modi Shi, Guanghui Ren, Maoqing Yao, Bin Zhao, et al. Hume: Introducing system-2 thinking in visual-language-action model. *arXiv preprint arXiv:2505.21432*, 2025.

Van-Tuan Tran, Shruthi Gowda, Merim Dzaferagic, and Marco Ruffini. Is our benchmark enough? an analysis of continual learning for mllms. *arXiv preprint arXiv:2606.20961*, 2026.

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, May 2024. doi: 10.1109/icra57147.2024.10611129.

Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(8):5362–5383, August 2024a. ISSN 1939-3539. doi: 10.1109/tpami.2024.3367329.

Shaowen Wang, Linxi Yu, and Jian Li. Lora-ga: Low-rank adaptation with gradient approximation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 54905–54931. Curran Associates, Inc., 2024b. doi: 10.52202/079017-1741.

Xiao Wang, Tianze Chen, Qiming Ge, Han Xia, Rong Bao, Rui Zheng, Qi Zhang, Tao Gui, and Xuanjing Huang. Orthogonal subspace learning for language model continual learning. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 10658–10671, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.715.

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, 2026a.

Yihao Wang, Pengxiang Ding, Lingxiao Li, Can Cui, Zirui Ge, Xinyang Tong, Wenxuan Song, Han Zhao, Wei Zhao, Pengxu Hou, et al. Vla-adapter: An effective paradigm for tiny-scale vision-language-action model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40 (22):18638–18646, March 2026b. ISSN 2159-5399. doi: 10.1609/aaai.v40i22.38931.

Yuxuan Wu, Guangming Wang, Zhiheng Yang, Tianchen Deng, Maoqing Yao, Brian Sheil, and Hesheng Wang. Continually evolving skill knowledge in vision language action model. *arXiv preprint arXiv:2511.18085*, 2026.

Fanqi Yu, Matteo Tiezzi, Tommaso Apicella, Cigdem Beyan, and Vittorio Murino. Lifelong imitation learning with multimodal latent replay and incremental adjustment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6740–6749, June 2026.

Yahan Yu, Duzhen Zhang, Yong Ren, Xuanle Zhao, Xiuyi Chen, and Chenhui Chu. Progressive LoRA for multimodal continual instruction tuning. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 2779–2796, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-256-5. doi: 10.18653/v1/2025.findings-acl.143.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. In Pulkit Agrawal, Oliver Kroemer, and Wolfram Burgard (eds.), *Proceedings of The 8th Conference on Robot Learning*, volume 270 of *Proceedings of Machine Learning Research*, pp. 3157–3181. PMLR, 06–09 Nov 2025.

Jianke Zhang, Xiaoyu Chen, Yanjiang Guo, Yucheng Hu, and Jianyu Chen. Vlm4vla: Revisiting vision-language-models in vision-language-action models. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 11357–11376, 2026a.

Likui Zhang, Tao Tang, Zhihao Zhan, Xiuwei Chen, Zisheng Chen, Jianhua Han, Jiangtong Zhu, Pei Xu, Hang Xu, Hefeng Wu, et al. Atomicvla: Unlocking the potential of atomic skill learning in robots. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20743–20754, June 2026b.

Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In Jie Tan, Marc Toussaint, and Kourosh Darvish (eds.), *Proceedings of The 7th Conference on Robot Learning*, volume 229 of *Proceedings of Machine Learning Research*, pp. 2165–2183. PMLR, 06–09 Nov 2023.

ThoughtVLA: Evolving Thought Trajectories for Continual Vision-Language-Action Adaptation

Supplementary Material

Overview

This supplementary material presents the theoretical analysis, implementation details, dataset examples, and additional results for ThoughtVLA. Notation and evaluation metrics follow the main paper.

(1) Theoretical analysis (Sec. A). We establish the low-rank approximation underlying CKT with activations held constant and explain how its score measures complementary transfer.

(2) Implementation details (Sec. B). We specify ESR and CKT, stage-wise training, persistent state, and evaluation protocols, with task lists and representative simulation demonstrations.

(3) Additional results (Sec. C). We provide cross-backbone comparisons, resource measurements, and task-order and training-seed analyses, together with ablation controls.

(4) Limitations (Sec. D). We discuss protocol assumptions, library growth, and the scope of the theoretical and experimental evidence.

A Theoretical Analysis

A.1 CKT Analysis with Constant Activations

The analysis justifies CKT's score in Eq. (8) as the reduction in response error achievable after rank-r adaptation, with activations, gradients, and normalization constants held constant.

Proposition A.1 (Attainable low-rank response)**.** *Let $\mathbf{H} \in \mathbb{R}^{d_{\text{in}} \times m}$ be an activation matrix, $\mathbf{V}, \mathbf{D} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ be a target parameter change and a candidate transfer, and $c > 0$. For a nonnegative integer rank budget r, define $\mathbf{R} = (\mathbf{V} - \mathbf{D})\mathbf{H}/c$. Then*

$$\min_{\operatorname{rank}(\Delta \mathbf{W}) \le r} \|\mathbf{R} - \Delta \mathbf{W}\mathbf{H}/c\|_F^2 = \epsilon_r(\mathbf{R}) = \sum_{a>r} \sigma_a^2(\mathbf{R}). \tag{A1}$$

The update has no additional norm, sparsity, or factor-sharing constraints. The activation matrix need not have full rank.

Proof. Every feasible $\Delta \mathbf{W}$ satisfies $\operatorname{rank}(\Delta \mathbf{W}\mathbf{H}/c) \le r$. The Eckart–Young–Mirsky identity used in Eq. (8) therefore gives the lower bound

$$\|\mathbf{R} - \Delta \mathbf{W}\mathbf{H}/c\|_F^2 \ge \epsilon_r(\mathbf{R}). \tag{A2}$$

To establish attainability through $\mathbf{H}$, let $\mathbf{R}_r$ be a rank-at-most-r truncated singular-value decomposition of $\mathbf{R}$, retaining only nonzero singular components. Because $\mathbf{R}$ is a left multiple of $\mathbf{H}$, its row space is contained in the row space of $\mathbf{H}$. The rows of $\mathbf{R}_r$ remain in that space. Thus, with $\mathbf{H}^\dagger$ the Moore–Penrose pseudoinverse,

$$\mathbf{R}_r \mathbf{H}^\dagger \mathbf{H} = \mathbf{R}_r, \qquad \Delta \mathbf{W}^\star = c \mathbf{R}_r \mathbf{H}^\dagger, \tag{A3}$$

$$\operatorname{rank}(\Delta \mathbf{W}^\star) \le \operatorname{rank}(\mathbf{R}_r) \le r, \qquad \Delta \mathbf{W}^\star \mathbf{H}/c = \mathbf{R}_r. \tag{A4}$$

This update attains error $\|\mathbf{R} - \mathbf{R}_r\|_F^2 = \epsilon_r(\mathbf{R})$, establishing equality. Any such rank-at-most-r matrix admits a product of two LoRA factors of inner dimension r, with zero padding if its rank is smaller. □

Corollary A.2 (Transfer score as local error reduction)**.** *For skill $q \in \mathcal{A}_s$, fix the compressed activations $\overline{\mathbf{H}}_q^v$, positive constants c_v, and target responses $\mathbf{Y}_q^v$ in Eq. (7). Let Ω_r contain independent*

rank-at-most-r updates for all adapted layers, and let $w_v \geq 0$ with $\sum_v w_v = 1$. The minimum weighted response error after candidate transfer $\mathcal{F}$ is

$$J_r(q, \mathcal{F}) = \min_{\{\Delta \mathbf{W}^v\}_v \in \Omega_r} \sum_v w_v \left\| \mathbf{Y}_q^v - (\mathbf{D}_{\mathcal{F}}^v + \Delta \mathbf{W}^v) \overline{\mathbf{H}}_q^v / c_v \right\|_F^2 \tag{A5}$$

$$= \sum_v w_v \, \epsilon_r(\mathbf{R}_{q,\mathcal{F}}^v), \tag{A6}$$

$$\Delta_r(q, \mathcal{F}) = J_r(q, \varnothing) - J_r(q, \mathcal{F}). \tag{A7}$$

Consequently, maximizing $\Delta_r(q, \mathcal{F})$ over the candidate set minimizes this local approximation error over the same set. A positive score gives strictly lower error than no transfer.

Proof. The target change at layer v is $\mathbf{V}^v = -\gamma_{\text{geo}} \nabla_{\mathbf{W}^v} \mathcal{E}_q^{\text{write}}$. Hence $\mathbf{R}_{q,\mathcal{F}}^v = (\mathbf{V}^v - \mathbf{D}_{\mathcal{F}}^v) \overline{\mathbf{H}}_q^v / c_v$ satisfies Proposition A.1. With activations held constant and independently constrained layer updates, each term is minimized separately. Applying the proposition and summing the minima gives Eq. (A6). For no transfer, $\mathbf{D}_{\varnothing}^v = \mathbf{0}$ and $\mathbf{R}_{q,\varnothing}^v = \mathbf{Y}_q^v$. Subtraction yields Eq. (A7) and the stated ordering. □

Why residual magnitude is insufficient. A smaller residual need not leave a smaller error after low-rank adaptation. Consider one layer with $r = 1$, $\mathbf{H} = \mathbf{I}_2$, $c = 1$, and the target response and two candidate residuals

$$\mathbf{Y} = \begin{pmatrix} 4 & 0 \\ 0 & 2 \end{pmatrix}, \qquad \mathbf{R}_A = \begin{pmatrix} 5 & 0 \\ 0 & 0.5 \end{pmatrix}, \qquad \mathbf{R}_B = \begin{pmatrix} 1.5 & 0 \\ 0 & 1.5 \end{pmatrix}. \tag{A8}$$

The transfers are $\mathbf{D}_A = \mathbf{Y} - \mathbf{R}_A$ and $\mathbf{D}_B = \mathbf{Y} - \mathbf{R}_B$. Although $\|\mathbf{R}_B\|_F^2 = 4.5$ is below $\|\mathbf{R}_A\|_F^2 = 25.25$, the rank-one approximation errors satisfy $\epsilon_1(\mathbf{R}_A) = 0.25 < 2.25 = \epsilon_1(\mathbf{R}_B)$. The no-transfer error is $\epsilon_1(\mathbf{Y}) = 4$, so CKT assigns $\Delta_1(A) = 3.75$ and $\Delta_1(B) = 1.75$. Candidate A leaves a larger response that is more closely approximated within the new update's rank budget. Candidate transfers may contain multiple retained updates and are not constrained to the new update's rank.

B Implementation Details

B.1 Stage-Wise Learning

All new skills $q \in \mathcal{A}_s$ use the same pre-stage snapshot $(\boldsymbol{\theta}_{s-1}, \mathcal{H}_{s-1}, \mathcal{K}_{s-1}, \mathcal{I}_{s-1}^{\text{CKT}})$. Initially, $\mathcal{K}_0 = \mathcal{I}_0^{\text{CKT}} = \varnothing$ and $\mathcal{H}_0$ has no records and zero accumulated statistics, so stage 1 learns without transfer. For each skill q, demonstrations are divided into $\mathcal{B}_q^{\text{write}}$, $\mathcal{B}_q^{\text{fit}}$, $\mathcal{B}_q^{\text{val}}$, $\mathcal{B}_q^{\text{train}}$, and a separate test set. The corresponding stage subsets are $\mathcal{B}_s^a = \bigcup_{q \in \mathcal{A}_s} \mathcal{B}_q^a$ for $a \in \{\text{write}, \text{fit}, \text{val}, \text{train}\}$. ESR builds each skill's task knowledge from its write subset and forms its attribute profile from all training instructions for that skill. CKT uses each skill's write, fit, and validation subsets for transfer scoring and confirmation, then the stage training subset to learn the shared new LoRA. All samples from a trajectory stay in one subset; test data do not enter selection or adaptation. Algorithm 1 summarizes stage-wise learning and inference using the notation of the main paper. Each candidate $\mathcal{I}_{s-1}^{\text{CKT}}(k)$ retains the previous skill's complete LoRA composition, including its update references and coefficients; $\mathcal{F}_q^\star = \varnothing$ denotes no transfer for skill q.

Algorithm 1 ThoughtVLA: continual skill acquisition and pre-action skill reasoning.

Require: Frozen base VLA $\boldsymbol{\theta}_0$ with semantic encoder φ_0; LoRA rank r
Require: Stage demonstrations $\{\mathcal{B}_s\}_{s=1}^{S}$ grouped by skills $\mathcal{A}_s$
Initialize $\mathcal{K}_0 = \mathcal{I}_0^{\mathrm{CKT}} = \varnothing$ and $\mathcal{H}_0 = (\varnothing, \mathbf{0})$
for stage $s = 1, \ldots, S$ **do**
  Form write, fit, validation, and training subsets for each $q \in \mathcal{A}_s$
  for skill $q \in \mathcal{A}_s$ **do**
    $\mathcal{F}_q^\star \leftarrow \varnothing$
    if $\mathcal{K}_{s-1} \neq \varnothing$ **then**
      Score compositions $\mathcal{I}_{s-1}^{\mathrm{CKT}}(k)$, $k \in \mathcal{K}_{s-1}$, by $\Delta_r(q, \cdot)$ on $\mathcal{B}_q^{\mathrm{write}}$
      $k_q^\star \leftarrow \arg\max_{k \in \mathcal{K}_{s-1}} \Delta_r(q, \mathcal{I}_{s-1}^{\mathrm{CKT}}(k))$
      if $\Delta_r(q, \mathcal{I}_{s-1}^{\mathrm{CKT}}(k_q^\star)) > 0$ **then**
        Run paired transfer and no-transfer trials on $\mathcal{B}_q^{\mathrm{fit}}$
        if validation on $\mathcal{B}_q^{\mathrm{val}}$ accepts the proposal (Sec. B.3) **then**
          $\mathcal{F}_q^\star \leftarrow \mathcal{I}_{s-1}^{\mathrm{CKT}}(k_q^\star)$
        end if
        Discard both trial LoRAs ▷ No retry after rejection
      end if
    end if
  end for
  Train shared $\boldsymbol{\omega}_s$ from fresh initialization on $\mathcal{B}_s^{\mathrm{train}}$ using $\{\mathcal{F}_q^\star\}_{q \in \mathcal{A}_s}$
  Freeze and retain $\boldsymbol{\omega}_s$ ▷ Shared stage update; Eq. (A12)
  Retain previous skills' execution maps in $\mathcal{I}_s^{\mathrm{CKT}}$
  for skill $q \in \mathcal{A}_s$ **do**
    $\mathcal{I}_s^{\mathrm{CKT}}(q) \leftarrow \mathcal{F}_q^\star \cup \{(\Delta \mathbf{W}_s^v, 1)\}_v$
    Construct task knowledge $\boldsymbol{\eta}_q$, including statistics $\boldsymbol{\xi}_q$
  end for
  $\mathcal{K}_s \leftarrow \mathcal{K}_{s-1} \cup \mathcal{A}_s$; update $\mathbf{S}_s$ and $\mathcal{H}_s$ by Eq. (6)
end for
Inference after stage s:
Receive initial images and instruction $\boldsymbol{\chi} = (\mathbf{x}_0, \ell)$
Run ESR once: $(\boldsymbol{\tau}, q) \leftarrow G_s(\boldsymbol{\chi})$ ▷ $\mathbf{c}_s^0 \to \mathbf{c}_s^1 \to \mathbf{c}_s^2 \to q$
Configure the action head with $\mathcal{I}_s^{\mathrm{CKT}}(q)$ using Eq. (5)
Generate $\mathbf{u}_h = F_{\boldsymbol{\theta}_s, \mathcal{I}_s^{\mathrm{CKT}}}(\mathbf{x}_h, \ell, q)$ from current observations
  throughout the episode, retaining the selected composition

B.2 ESR Knowledge and Reasoning Traces

For skill q, the retained tuple $\boldsymbol{\eta}_q = (\boldsymbol{\mu}_q, \mathbf{p}_q^{\mathrm{lex}}, \mathbf{p}_q^{\mathrm{attr}}, \boldsymbol{\xi}_q)$ comprises its semantic representation, word and character counts, task attributes, and contributions to normalization and weighting. Within this tuple, $\boldsymbol{\xi}_q$ stores sample counts, semantic feature moments, and document frequencies. These statistics accumulate according to Eq. (6), with $\mathbf{S}_0 = \mathbf{0}$. The skill-indexed vectors $\mathbf{e}_s^{\mathrm{lex}}$ and $\mathbf{e}_s^{\mathrm{attr}}$ contain lexical evidence and unstandardized task-attribute scores, respectively. We obtain $\boldsymbol{\mu}_q$ by averaging normalized VLM features from the jointly encoded images and instructions in $\mathcal{B}_q^{\mathrm{write}}$. Let $\mathbf{a}_s$ and $\mathbf{v}_s$ be the accumulated feature mean and coordinate-wise variance computed from $\mathbf{S}_s$. The operator Norm performs ℓ_2 normalization, α_c controls centering, p controls variance scaling, and $\varepsilon > 0$ prevents division by zero. For an input feature or a skill-associated mean $\mathbf{z}$, normalization uses $\mathsf{N}_s(\mathbf{z}) = \mathrm{Norm}((\mathbf{z} - \alpha_c \mathbf{a}_s)/(\mathbf{v}_s + \varepsilon)^{p/2})$, where division and powers are coordinate-wise, and α_c, p are stage-independent. The moments are accumulated before this shared transformation.

Instruction matching supplies the intermediate scores used before explicit attribute checking. ESR weights word and character n-grams by their frequencies in the current instruction and their accumulated document frequencies, then compares the weighted instruction with each candidate's lexical profile. The resulting evidence $\mathbf{e}_s^{\mathrm{lex}}$ updates the initial scores as

$$\mathbf{c}_s^1 = \mathbf{c}_s^0 + g_s^{\mathrm{lex}} \lambda_{\mathrm{lex}} \mathbf{e}_s^{\mathrm{lex}}. \tag{A9}$$

The stage-independent weight λ_{lex} scales this correction. The gate g_s^{lex} is one only when the two highest scores in $\mathbf{c}_s^0$ differ by at most ϵ_{lex} and the range of $\mathbf{e}_s^{\text{lex}}$ within $\mathcal{C}_s$ reaches a specified minimum. Otherwise the initial scores are retained and attribute checking is skipped.

Task-attribute agreement. Where semantic and instruction-matching evidence leaves the task interpretation ambiguous, ESR checks task attributes against the candidate skills. The initial semantic score jointly encodes the image and instruction; for the explicit check, a predefined parser identifies available attributes in the instruction without task or stage IDs. The attributes comprise spatial relation (left/right, above/below, inside/outside, front/behind), manner (horizontal/vertical, clockwise/counterclockwise), direction, and count (two through five or multiplicity inferred from repeated content words). The task knowledge retains these categories separately. For each value, $\mathbf{p}_q^{\text{attr}}$ stores its frequency among all training instructions for skill q, including instructions in which the value is absent. If $\mathcal{T}(\ell)$ is the set of values observed in the current instruction, the raw agreement is

$$[\mathbf{e}_s^{\text{attr}}]_q = \frac{1}{|\mathcal{T}(\ell)|} \sum_{a \in \mathcal{T}(\ell)} [\mathbf{p}_q^{\text{attr}}]_a, \qquad \mathcal{T}(\ell) \neq \varnothing. \tag{A10}$$

Observed values are deduplicated; conflicting values remain in the set rather than being resolved using a task label. Without an observed value, attribute checking is skipped. ESR centers the raw agreements within $\mathcal{C}_s$ and divides by their candidate-set standard deviation plus ε to obtain $\bar{\mathbf{e}}_s^{\text{attr}}$. The shared gate g_s^{attr} is one only after initial correction, when $\mathcal{T}(\ell) \neq \varnothing$, the two highest scores in $\mathbf{c}_s^1$ remain within a specified margin, and the range of raw attribute agreement within $\mathcal{C}_s$ reaches a specified minimum. Equation (3) then revises their scores using the stated weight λ_{attr}; otherwise $\mathbf{c}_s^2 = \mathbf{c}_s^1$. A skipped refinement leaves its trace entry empty and the scores unchanged; consulted evidence is retained even when it produces no revision. We use $K_c = 5$, $\alpha_c = 0.5$, $p = 1$, $\lambda_{\text{lex}} = 0.05$, and $\lambda_{\text{attr}} = 0.01$; both score-gap margins are 0.025, while the minimum lexical and attribute evidence ranges are 0.025 and 0.05, respectively. Evidence outside $\mathcal{C}_s$ is zero, so other skills retain their initial scores. As new records enter the skill library, shared normalization and instruction weighting can change the evidence used for subsequent traces without changing the pretrained VLM or the reasoning rules.

B.3 CKT Candidate Scoring, Confirmation, and Training

For a previous skill $k \in \mathcal{K}_{s-1}$, transfer uses the complete skillwise LoRA composition recorded in $\mathcal{I}_{s-1}^{\text{CKT}}(k)$. For each new skill q, geometry uses the frozen base policy and the same sample of $2r$ activation columns per layer (32 at $r = 16$). Candidates for q share the per-layer response normalization, gradient scale, data, gradients, activations, and random quantities. CKT scores each previous-skill candidate by $\Delta_r(q, \mathcal{F})$ in Eq. (8). It proposes a candidate maximizing that score only when the maximum is positive; otherwise it uses no transfer. Only the proposed candidate enters paired adaptation trials.

To assess the proposed transfer under action-loss training, the selected candidate and no-transfer branch undergo the same number of trial steps on $\mathcal{B}_q^{\text{fit}}$, sharing initialization, batches, and random quantities. The transfer trial uses the half-minibatch mask in Eq. (A12), while the no-transfer trial keeps previous updates inactive. For validation, the transfer branch activates all selected updates as at inference and is compared with the no-transfer branch using their respective trial LoRAs. Losses are normalized within each trajectory and averaged equally across at least three independent validation trajectories. Let L_t^{tr} and L_t^{no} be the held-out action losses of the transfer and no-transfer trial LoRAs on trajectory t. Their relative gain is

$$g_t^{\text{val}} = \frac{L_t^{\text{no}} - L_t^{\text{tr}}}{\max(L_t^{\text{no}}, \varepsilon)}. \tag{A11}$$

Let n_{val} be the number of validation trajectories, and $\delta_{\text{val}} = 0.01$ the acceptance hyperparameter. Acceptance requires $\text{mean}_t\, g_t^{\text{val}} \geq \delta_{\text{val}}$ and $\sum_t \mathbf{1}[g_t^{\text{val}} > 0] > n_{\text{val}}/2$. Unsuccessful confirmation or insufficient validation trajectories return no transfer without testing another candidate. Confirmation fixes the transfer choice; both trial LoRAs are then discarded before fresh initialization and final training on $\mathcal{B}_s^{\text{train}}$. Different skills may retain different previous updates even though they share the new stage LoRA.

Final stage training. With previous updates frozen, CKT trains a new action-head update $\boldsymbol{\omega}_s = \{\Delta \mathbf{W}_s^v\}_v \in \Omega_r$, where Ω_r contains the layer-update collections with rank at most r per layer, parameterized by LoRA factors. Training on $\mathcal{B}_s^{\text{train}}$ minimizes

$$\mathcal{E}_s^{\text{train}}(\boldsymbol{\omega}_s) = \mathbb{E}_{\mathcal{M}\sim\mathcal{B}_s^{\text{train}}}\left[\frac{1}{B}\sum_{i=1}^{B}\mathcal{L}_{\text{act}}\left(F_{\boldsymbol{\theta}_{s-1},\mathcal{I}_{s-1}^{\text{CKT}};\mathcal{F}_{q_i}(m_i),\boldsymbol{\omega}_s}(\mathbf{x}_i,\ell_i,q_i),\mathbf{u}_i\right)\right]. \tag{A12}$$

Here, $\mathcal{M} = \{(\mathbf{x}_i,\ell_i,q_i,\mathbf{u}_i)\}_{i=1}^B$ is a minibatch and the binary mask m_i satisfies $\sum_i m_i = B/2$, where $q_i \in \mathcal{A}_s$ indexes the sample's skill, $\mathcal{F}_{q_i}(1) = \mathcal{F}_{q_i}^\star$, and $\mathcal{F}_{q_i}(0) = \varnothing$.

B.4 Persistent State and Resource Accounting

Persistent state comprises LoRA updates in $\boldsymbol{\theta}_s$, task records and shared statistics in $\mathcal{H}_s$, and update references and coefficients in $\mathcal{I}_s^{\text{CKT}}$.

B.5 Datasets and Continual Streams

Each manipulation task supplies demonstrations for one skill; stage boundaries and training groupings are known. After each stage, evaluation covers all manipulation tasks encountered so far without oracle skill or stage IDs, except for the oracle diagnostic in Table 6. Table A1 specifies the stage partitions.

Table A1: Continual streams. The task count refers to manipulation tasks, each associated with one learned skill. The stage split gives the initial count followed by the number introduced per subsequent stage.

Setting	Tasks	Stages	Stage split
LIBERO-Object / Goal / Spatial / Long	10 each	6 each	$5+1\times5$
RoboTwin 2.0	50	6	$25+5\times5$
GR-1 Humanoid	24	7	$12+2\times6$
Mixed-embodiment	55	51	$5+1\times50$
Cross-backbone (four LIBERO streams)	10 each	6 each	$5+1\times5$
Real-robot (UR-10e)	20	6	$10+2\times5$

Single-arm manipulation. The four LIBERO suites form independent streams. Object varies the object to be manipulated, Goal varies the requested outcome, Spatial varies spatial relationships, and Long includes multi-step manipulation. Tables A4 and A5 list their instructions and order. Each stream starts from the same pretrained configuration rather than inheriting updates from another suite.

Bimanual and humanoid manipulation. RoboTwin 2.0 comprises 50 bimanual manipulation tasks, including handovers, coordinated lifting, object placement, tool use, and stacking. Evaluation uses instruction formulations absent from adaptation, while the corresponding skills have already been learned. Table A6 lists representative adaptation instructions, not the held-out evaluation formulations. The RoboCasa-based GR-1 Humanoid setting comprises 24 manipulation tasks involving object relocation, receptacles, and articulated fixtures (Table A7). Simulation success follows the corresponding environment's task predicate.

Mixed-embodiment and cross-backbone evaluation. The mixed stream introduces 30 single-arm, 15 bimanual, and 10 humanoid manipulation tasks in that order. Its mapping to the source task lists is given in Table A8. Cross-backbone evaluation instead repeats the four independent LIBERO streams with Qwen3.5-2B and an OpenVLA-OFT-style action head, retaining their task order and stage partitions.

Demonstration examples. Figures A1–A4 illustrate the simulation and real-robot demonstrations. Each row samples five frames uniformly from one trajectory, including its first and last frames; percentages indicate relative trajectory time.

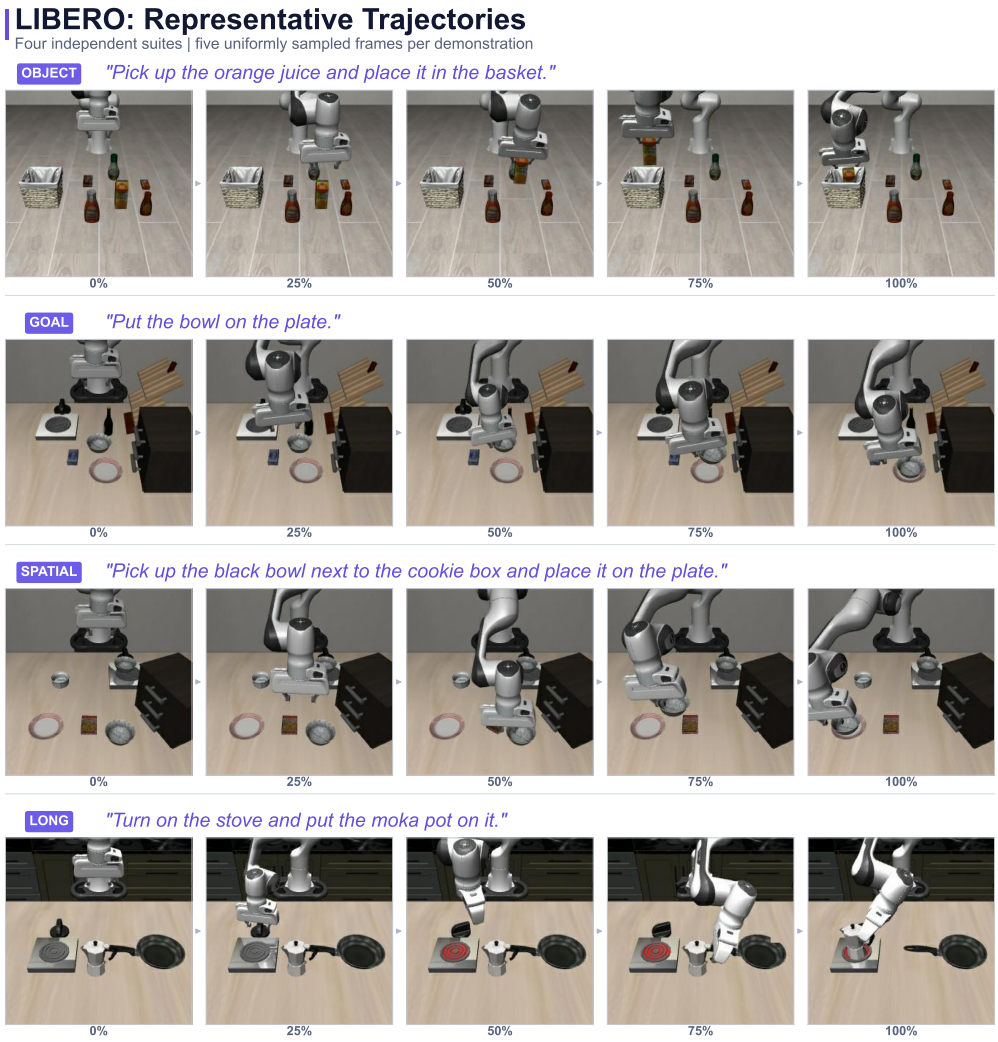

Figure A1: **LIBERO demonstration examples.** One trajectory from each of Object, Goal, Spatial, and Long, with five uniformly sampled frames per row. Source: the Hugging Face repository `physical-intelligence/libero` (CC BY 4.0), a LeRobot conversion of OpenVLA's modified LIBERO release.



Figure A2: **RoboTwin 2.0 demonstration examples.** Five bimanual tasks: dual-bottle lifting, pot lifting, scanning, laptop opening, and block stacking. Each row shows five uniformly sampled frames. Source: the Hugging Face repository `TianxingChen/RoboTwin2.0` (MIT license).

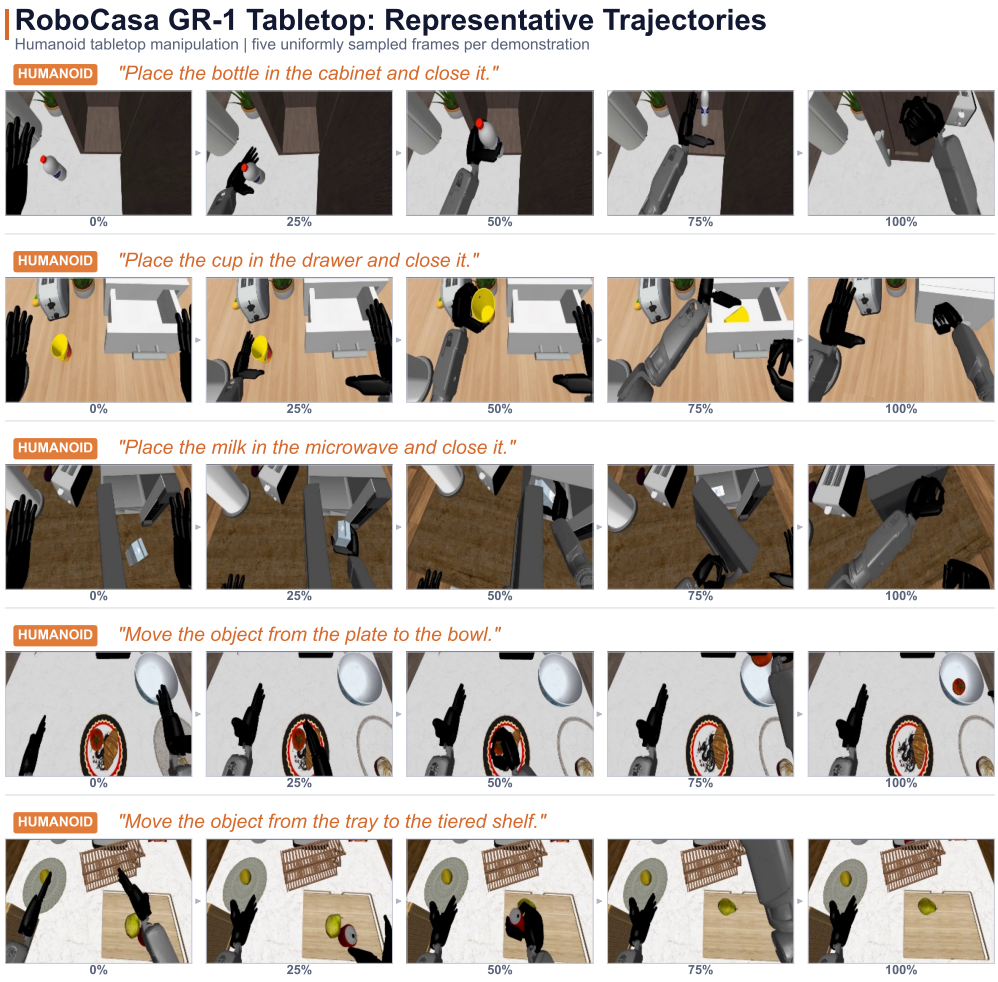

Figure A3: **RoboCasa GR-1 Tabletop demonstration examples.** Five humanoid tasks involving receptacles and object relocation, with five uniformly sampled frames per row. Source: NVIDIA's Hugging Face dataset `nvidia/PhysicalAI-Robotics-GR00T-X-Embodiment-Sim` (CC BY 4.0).

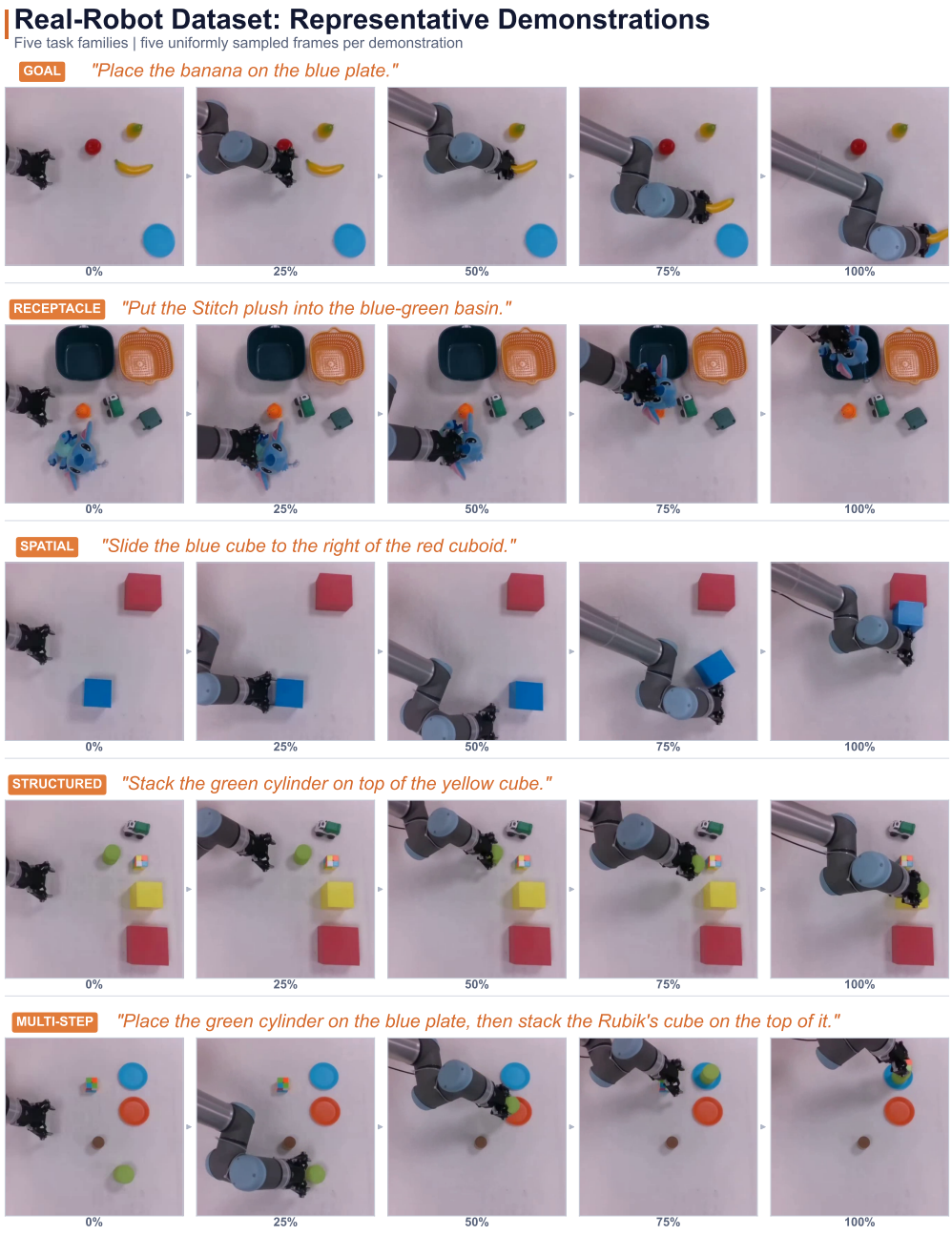

Figure A4: **Real-robot demonstration examples.** One task from each family: goal-directed manipulation, receptacle interaction, spatial-relation reasoning, structured manipulation, and multi-step manipulation. Each row shows five uniformly sampled frames from a dataset demonstration.

B.6 Training and Evaluation Protocol

The primary policy initializes Qwen3-VL-4B and a GR00T-style flow-matching DiT action head from a StarVLA VLA pretrained on Open X-Embodiment (Community, 2026; Collaboration et al., 2025). ThoughtVLA freezes the pretrained base weights and previous LoRA updates, learning one action-head LoRA set per stage with rank 16 and scaling factor 32. No previous raw demonstration trajectories are replayed. All methods are implemented in the StarVLA codebase with shared data processing, optimization, and evaluation interfaces; their adaptation and replay mechanisms follow the respective methods.

Images are resized to 224×224, with a maximum text length of 2,048. The action head uses four denoising steps. Training uses AdamW with learning rate 10^{-4}, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 10^{-8}, and gradient clipping at 1.0. A cosine schedule follows 2,000 warmup steps, reaching 10^{-6} for LIBERO and the mixed stream, and 5×10^{-7} for RoboTwin and GR-1. The stage-1/later-stage training budgets are 80K/6K steps for LIBERO, 100K/20K for RoboTwin, 100K/10K for GR-1, and 40K/10K for real-robot experiments. Batch size is 16 for RoboTwin and 24 for the other three settings. Training uses BF16 and DeepSpeed ZeRO-2 on two NVIDIA A100 GPUs, with gradient checkpointing and no gradient accumulation. The reference training seed is 42; the two additional seeds in Table A12 are 43 and 44. LIBERO and RoboTwin evaluation seeds are 7 and 0.

For the three simulated embodiments, action normalization uses separate statistics computed from each embodiment's training data when it first enters the stream; each set is retained without later recalculation. Each embodiment uses its own action-normalization statistics; no shared set is used across embodiments. No future-stage data or action-normalization statistics are accessed. The action policy uses the observation interface reported in Table A3. After each stage, every encountered manipulation task is evaluated with 50 rollouts on LIBERO and 100 rollouts on RoboTwin 2.0 and GR-1 Humanoid. Mixed-embodiment tasks retain the rollout count of their corresponding simulator. A successful rollout scores 100 and a failed rollout scores 0; task success is their mean.

Mixed-embodiment interface. The mixed stream uses a 58D state, 29D action, and 50-step horizon. LIBERO's 7D state and 7D action with an eight-step horizon and RoboTwin's 14D state and 14D action with a 50-step horizon are padded to this layout; GR-1 retains its 58D state and 29D action while its 16-step horizon is padded to 50. Valid channels retain their source coordinate order and use their embodiment's retained action normalization. Masks exclude padded channels and time steps from the action objective. Embodiment-specific experiments retain their native interfaces (Table A3).

Comparison methods. Following the implementation designs in their respective papers, all comparison methods are implemented with trainable LoRA adapters in both the VLM backbone and action head. ThoughtVLA instead trains LoRA adapters only in the action head and keeps the VLM frozen. Sequential FT updates the policy across stages without an explicit retention mechanism. Among low-rank methods, O-LoRA constrains interference through orthogonal update subspaces (Wang et al., 2023), while ProgLoRA progressively expands adaptation modules (Yu et al., 2025). CL-MoE uses a dual-momentum mixture of experts for continual multimodal learning (Huai et al., 2025); these methods provide comparisons beyond approaches originally developed for robot control.

For continual robot learning, SkillsCrafter learns language-conditioned manipulation skills (Wang et al., 2026a), and DMPEL grows a parameter-efficient expert library with learned mixture coefficients (Lei et al., 2026). AtomicVLA predicts atomic-skill sequences and routes execution through an expandable expert library (Zhang et al., 2026b). CLARE combines adapter expansion with autonomous routing for continual VLA adaptation (Römer et al., 2026). MLR-IFA instead combines multimodal latent replay with incremental adjustment (Yu et al., 2026).

Metrics and comparisons. Let S be the final stage, $\mathcal{K}_s$ the skills learned by stage s, $a_{s,q}$ the success rate on skill q's manipulation task after stage s, and s_q the stage when q is learned. We compute $\mathrm{FSR} = |\mathcal{K}_S|^{-1} \sum_{q \in \mathcal{K}_S} a_{S,q}$ and $\mathrm{CSR} = S^{-1} \sum_{s=1}^{S} |\mathcal{K}_s|^{-1} \sum_{q \in \mathcal{K}_s} a_{s,q}$. Forgetting clips each decline from post-learning success at zero, averages over later stages for each skill, and then averages previous skills: $\mathrm{Forg.} = |\mathcal{K}_{S-1}|^{-1} \sum_{q \in \mathcal{K}_{S-1}} (S - s_q)^{-1} \sum_{t=s_q+1}^{S} \max\{0, a_{s_q,q} - a_{t,q}\}$. Success rates use a 0–100 scale and forgetting uses percentage points; ID and OOD are computed separately. Table A2 separates matched experimental conditions from method-specific adaptation and replay.

Table A2: **Fair-comparison protocol for Tables 1–4.** Shared conditions apply within each setting; adaptation and replay follow each comparison method.

Comparison axis	ThoughtVLA	Comparison methods
Shared protocol within each setting		
Initial VLA	StarVLA VLA weights pretrained on Open X-Embodiment; Qwen3-VL-4B with a GR00T-style DiT head in the primary setting and Qwen3.5-2B with an OpenVLA-OFT-style head in the cross-backbone setting	Same setting-specific pretrained VLA initialization
Task stream and demonstrations	Shared stage grouping, task order, and current-stage demonstrations	Same tasks, order, and incoming demonstrations
Input and action processing	StarVLA image, language, and action processing for each embodiment	Same processing and action interface
Optimization	AdamW, learning rate, schedule, warmup, and batch size specified above	Same setting-specific configuration
Formal adaptation budget	Stage-wise optimizer steps specified above	Same formal steps per stage
Evaluation	All encountered tasks after each stage, with shared rollout initializations and no oracle skill or stage ID	Same evaluation inputs, initializations, and rollout counts
Metrics	FSR, CSR, and Forg. as defined above	Same definitions and aggregation
Method-specific choices		
Trainable components	Frozen VLM and base action head; one new action-head LoRA set per stage	LoRA adaptation in the VLM and action head, following each comparison method
Adapter organization	Rank-16 action-head LoRA with scaling factor 32; previous updates remain frozen	Method-specific ranks, expert expansion, and combination rules
Previous-task replay	No previous raw demonstrations or training samples replayed	MLR-IFA replays 100 training samples per task, not complete demonstrations; DMPEL follows its original replay setting
Retained state	Action-head LoRA bank and accumulated task knowledge	Method-specific adapters, experts, routing state, or replay samples

Table A3: **Observation–action interfaces.** Stage splits are given in Table A1. Action dimension, robot-state dimension, and prediction horizon are reported separately.

Setting	Embodiment	Action dim.	State dim.	Horizon
LIBERO (4 streams)	Single-arm	7	7	8
RoboTwin 2.0	Bimanual	14	14	50
GR-1 Humanoid	Humanoid	29	58	16
Mixed-embodiment	Heterogeneous	29	58	50
Cross-backbone	Single-arm	7	7	8
Real-robot	Single-arm	7	7	8

B.7 Real-Robot Data and Evaluation

The real-robot setting uses a UR-10e, an AG-95 gripper, and a stationary third-person RealSense D435i camera. The 20 manipulation tasks have 50 teleoperated demonstrations each, totaling 1,000 trajectories. They cover goal-directed manipulation, receptacle interaction, spatial-relation reasoning, structured manipulation, and multi-step manipulation. Table A9 lists the instructions and stage assignments.

Given RGB observations and robot state, the action policy predicts eight-step chunks of 7D delta actions: 3D Cartesian displacement, 3D axis–angle orientation displacement, and a scalar gripper command. An operational-space pose controller executes the first three actions before receiving a new observation and replanning. After each stage, every encountered manipulation task is evaluated with 10 ID and 10 OOD rollouts. Success requires completion of the instructed goal within 100 control steps.

Task objects are initialized within predefined regions before each rollout. ID trials use object and target locations, distractors, illumination, and backgrounds represented in the demonstrations. OOD trials keep the instruction and success criterion unchanged while varying one factor: object position,

goal position, increased clutter, altered lighting, or an unseen background. The 10 OOD rollouts include two trials for each of these five changes.

Success and trial accounting. Each rollout scores 100 on success and 0 on failure for its instructed goal. Goal-directed tasks require the object to remain on the target surface; receptacle tasks require containment or retrieval with the specified endpoint. Spatial tasks require the instructed relative position or removal, and structured tasks require the specified drawer state or stable stacking or unstacking outcome. Multi-step tasks require all subgoals in order.

B.8 Ordered Manipulation Tasks

The following tables specify manipulation goals, their order, and their training-stage assignments. Row numbers identify the skill learned for each goal within a stream; they are not additional inputs to ESR. RoboTwin and GR-1 entries give representative instructions.

Table A4: LIBERO-Object and LIBERO-Goal. The two suites are independent streams. Tasks 1–5 form stage 1; tasks 6–10 arrive individually at stages 2–6.

Task	Stage	LIBERO-Object	LIBERO-Goal
1	1	Pick up the alphabet soup and place it in the basket.	Open the middle drawer of the cabinet.
2	1	Pick up the cream cheese and place it in the basket.	Put the bowl on the stove.
3	1	Pick up the salad dressing and place it in the basket.	Put the wine bottle on top of the cabinet.
4	1	Pick up the BBQ sauce and place it in the basket.	Open the top drawer and put the bowl inside.
5	1	Pick up the ketchup and place it in the basket.	Put the bowl on top of the cabinet.
6	2	Pick up the tomato sauce and place it in the basket.	Push the plate to the front of the stove.
7	3	Pick up the butter and place it in the basket.	Put the cream cheese in the bowl.
8	4	Pick up the milk and place it in the basket.	Turn on the stove.
9	5	Pick up the chocolate pudding and place it in the basket.	Put the bowl on the plate.
10	6	Pick up the orange juice and place it in the basket.	Put the wine bottle on the rack.

Table A5: LIBERO-Spatial and LIBERO-Long, with the same stage partition as the other LIBERO streams.

Task	Stage	LIBERO-Spatial	LIBERO-Long
1	1	Pick up the black bowl between the plate and the ramekin and place it on the plate.	Put both the alphabet soup and the tomato sauce in the basket.
2	1	Pick up the black bowl next to the ramekin and place it on the plate.	Put both the cream cheese box and the butter in the basket.
3	1	Pick up the black bowl from the table center and place it on the plate.	Turn on the stove and put the moka pot on it.
4	1	Pick up the black bowl on the cookie box and place it on the plate.	Put the black bowl in the bottom drawer of the cabinet and close it.
5	1	Pick up the black bowl in the top drawer of the wooden cabinet and place it on the plate.	Put the white mug on the left plate and the yellow-and-white mug on the right plate.
6	2	Pick up the black bowl on the ramekin and place it on the plate.	Pick up the book and place it in the back compartment of the caddy.

Continued on the next page.

Table A5 continued.

Task	Stage	LIBERO-Spatial	LIBERO-Long
7	3	Pick up the black bowl next to the cookie box and place it on the plate.	Put the white mug on the plate and put the chocolate pudding to the right of the plate.
8	4	Pick up the black bowl on the stove and place it on the plate.	Put both the alphabet soup and the cream cheese box in the basket.
9	5	Pick up the black bowl next to the plate and place it on the plate.	Put both moka pots on the stove.
10	6	Pick up the black bowl on the wooden cabinet and place it on the plate.	Put the yellow-and-white mug in the microwave and close it.

Table A6: RoboTwin 2.0: ordered manipulation tasks and representative adaptation instructions. The initial 25 tasks form stage 1, followed by five tasks per stage. Evaluation uses held-out instruction formulations for these learned skills.

Task	Stage	Representative instruction
1	1	Lift the long, thin bottle with the right arm and hold it upright.
2	1	Grasp the medium metal hammer with the right arm and strike the block.
3	1	Arrange the red, green, and blue blocks from left to right in that order.
4	1	Arrange the large, medium, and small blocks centrally from largest to smallest.
5	1	Press the centered button on the black alarm clock.
6	1	Press the top of the white-and-black bell with the right arm.
7	1	Pick up the table bin and empty the balls into the larger bin.
8	1	Grasp the smooth roller.
9	1	Grasp the red block with the left arm, hand it to the right arm, and place it on the blue pad.
10	1	Pick up the white microphone and hand it to the other arm.
11	1	Pick up the mug, rotate and replace it, then hang it on the metal rack.
12	1	Securely lift the medium metal pot with both arms.
13	1	Move the sauce can next to the two-handled pot.
14	1	Move the orange-capped pill bottle onto the pad.
15	1	Pick up the playing-card box and move it outward.
16	1	Move the black stapler onto the magenta mat.
17	1	Open the laptop lid with the right arm.
18	1	Pull open the microwave door with the left arm.
19	1	Simultaneously lift the yellow-capped and yellow-lidded bottles.
20	1	Pick up one drink bottle with each arm.
21	1	Place the coffee box to the left of the mouse.
22	1	Place the blue bell to the right of the red stapler.
23	1	Place the brown bread in the woven basket.
24	1	Place the square bread piece in the round skillet.
25	1	Place the burger and fries on the dark brown tray.
26	2	Place the Coca-Cola can in the yellow basket, then lift the basket with the other arm.
27	2	Place the two lime cans in the blue plastic storage box using both arms.
28	2	Place the smooth bowl-shaped container on the ceramic plate.
29	2	Put the two green shoes, with their toes pointing left, into the orange shoe box.
30	2	Place the light-blue cup on the wooden coaster.
31	3	Place the small fan on the blue mat, facing the robot.
32	3	Place the curved-top mouse on the gray mat.
33	3	Place the small black card box in the woven basket, then shift the basket slightly.
34	3	Place the black-base bell on the electronic scale.
35	3	Place the remote control on the rectangular display stand.
36	4	Place the metallic phone on the angled green phone stand.
37	4	Place the white shoe on the mat with the right arm.
38	4	Press the orange-button stapler to staple the papers.
39	4	Place the three specified bottles in the white-lined dustbin one at a time.
40	4	Open the wooden cabinet drawer and place the ergonomic mouse inside.
41	5	Pick up the payment sign and rotate its QR code toward the robot.
42	5	Hold the scanner with the left arm and the tea box with the right arm, then scan the box.
43	5	Pick up the plastic bottle, shake it, and place it down.

Continued on the next page.

Table A6 continued.

Task	Stage	Representative instruction
44	5	Lift the screw-cap bottle with the left arm and shake it horizontally.
45	5	Move the red, green, and blue blocks to the center, then stack blue on green and green on red.
46	6	Place the red block in the center and stack the green block on it.
47	6	Stack the three blue bowls from largest at the bottom to smallest at the top.
48	6	Pick up the light-blue bowl with the left arm and stack the shiny bowl above it with the right arm.
49	6	Use the left arm to press the black-base stamp onto the magenta pad.
50	6	Press the gray switch to activate it.

Table A7: GR-1 Humanoid: ordered manipulation tasks and representative instructions. Tasks 1–12 form stage 1; subsequent stages each introduce two tasks.

Task	Stage	Representative instruction
1	1	Pick up the bottled water, place it in the cabinet, and close the cabinet.
2	1	Pick up the can, place it in the drawer, and close the drawer.
3	1	Pick up the cup, place it in the drawer, and close the drawer.
4	1	Pick up the milk, place it in the microwave, and close the microwave.
5	1	Pick up the potato, place it in the microwave, and close the microwave.
6	1	Pick up the wine, place it in the cabinet, and close the cabinet.
7	1	Pick up the can from the cutting board and place it in the basket.
8	1	Pick up the can from the cutting board and place it in the cardboard box.
9	1	Pick up the bell pepper from the cutting board and place it in the pan.
10	1	Pick up the bell pepper from the cutting board and place it in the pot.
11	1	Pick up the squash from the cutting board and place it in the tiered basket.
12	1	Pick up the croissant from the placemat and place it in the basket.
13	2	Pick up the squash from the placemat and place it in the bowl.
14	2	Pick up the bell pepper from the placemat and place it on the plate.
15	3	Pick up the tomato from the placemat and place it on the tiered shelf.
16	3	Pick up the sweet potato from the plate and place it in the bowl.
17	4	Pick up the bell pepper from the plate and place it in the cardboard box.
18	4	Pick up the tomato from the plate and place it in the pan.
19	5	Pick up the bell pepper from one plate and place it on the other plate.
20	5	Pick up the tomato from the tray and place it in the cardboard box.
21	6	Pick up the pear from the tray and place it on the plate.
22	6	Pick up the lemon from the tray and place it in the pot.
23	7	Pick up the eggplant from the tray and place it in the tiered basket.
24	7	Pick up the bell pepper from the tray and place it on the tiered shelf.

Table A8: Mixed-embodiment stream. Local indices refer to the preceding task tables. The cross-backbone setting separately reuses each complete LIBERO stream with its original six-stage partition.

Stages	Global tasks	Source setting	Local tasks
1	1–5	LIBERO-Object	1–5 jointly
2–6	6–10	LIBERO-Object	6–10, one per stage
7–16	11–20	LIBERO-Goal	1–10, one per stage
17–26	21–30	LIBERO-Spatial	1–10, one per stage
27–41	31–45	RoboTwin 2.0	1–15, one per stage
42–51	46–55	GR-1 Humanoid	1–10, one per stage

Table A9: Real-robot manipulation tasks. R1: goal-directed manipulation; R2: receptacle interaction; R3: spatial-relation reasoning; R4: structured manipulation; R5: multi-step manipulation. The initial ten tasks are followed by five stages of two tasks.

Task	Stage	Family	Instruction
1	1	R4	Close the top green drawer.
2	1	R1	Place the banana on the blue plate.
3	1	R1	Place the Rubik's cube on the blue plate.
4	1	R1	Place the Coke on the blue plate.
5	1	R2	Put the orange into the yellow perforated basin.
6	1	R2	Put the avocado into the blue-green basin.
7	1	R2	Put the Stitch plush into the blue-green basin.
8	1	R2	Take the Pepsi can out of the yellow perforated basin and place it on the blue plate.
9	1	R3	Remove the Pepsi can in front of the Coke can.
10	1	R3	Move the bottle to the left of the Pepsi can.
11	2	R3	Slide the blue cube to the right of the red cuboid.
12	2	R3	Place the banana in front of the watermelon slice.
13	3	R3	Place the orange next to the avocado.
14	3	R4	Stack the Rubik's cube on top of the red cuboid.
15	4	R4	Stack the green cylinder on top of the yellow cube.
16	4	R4	Remove the Rubik's cube from the red cuboid and place it on the blue plate.
17	5	R5	Put Rubik's cube in the top green drawer, then close it.
18	5	R5	Place the green cylinder on the blue plate, then stack the Rubik's cube on the top of it.
19	6	R5	Put the avocado in the yellow perforated basin, then put the Stitch plush into the same basin.
20	6	R5	Place the Coke can to the right of the Pepsi can, then place the bottle in front of the Coke can.

C Additional Results

C.1 Cross-Backbone and Resource Results

Reporting conventions. Comparisons use the same backbone, data, task order, and evaluation settings. All methods compute CSR by averaging success over all learned tasks at each stage, then averaging these stage means equally. Stream averages weight the four LIBERO suites equally.

Table A10: **Cross-backbone continual adaptation.** Results on four LIBERO streams with Qwen3.5-2B and an OpenVLA-OFT-style continuous-action head. Rates use a 0–100 scale. Shading identifies ThoughtVLA. Ranking marks follow Table 1.

Method	Object			Goal			Spatial			Long		
	FSR ↑	CSR ↑	Forg. ↓	FSR ↑	CSR ↑	Forg. ↓	FSR ↑	CSR ↑	Forg. ↓	FSR ↑	CSR ↑	Forg. ↓
Sequential FT	33.2	46.5	70.2	10.0	23.1	92.3	9.8	22.2	84.4	10.0	18.5	75.2
O-LoRA (EMNLP Findings'23)	49.0	58.3	42.4	43.8	42.9	50.9	50.8	54.7	31.8	10.8	20.6	57.6
CL-MoE (CVPR'25)	64.4	71.6	10.6	72.8	75.5	15.3	75.6	74.2	10.3	42.4	56.4	39.5
ProgLoRA (ACL Findings'25)	45.6	50.0	28.4	40.8	45.6	40.5	62.0	72.6	18.4	16.4	30.1	50.7
SkillsCrafter (AAAI'26)	88.8	90.7	6.6	91.6	92.9	4.6	70.8	71.3	10.3	40.8	65.4	40.1
DMPEL (TMLR'26)	91.4	91.9	6.0	77.8	80.5	8.6	71.8	72.0	10.5	57.2	70.3	19.3
AtomicVLA (CVPR'26)	88.6	90.2	7.1	87.4	89.1	6.2	66.2	69.0	7.8	50.0	62.5	20.6
MLR-IFA (CVPR'26)	89.4	93.5	8.4	83.2	84.9	13.6	78.0	77.7	5.6	58.2	69.1	33.3
CLARE (RA-L'26)	66.4	72.4	20.5	72.8	76.6	18.7	73.2	71.7	16.4	50.8	60.3	34.5
ThoughtVLA	**94.4**	**96.8**	**0.7**	**96.6**	**97.2**	**0.8**	**85.4**	**86.0**	**0.6**	**73.8**	**76.1**	**1.1**

Performance and resource cost. Table A11 pairs continual performance with the cost of retaining and using an expanding skill library on LIBERO-Object and RoboTwin 2.0. For ThoughtVLA, parameter growth counts the single action-head LoRA set added at each incremental stage; storage also includes task knowledge and skillwise LoRA composition records. Training time includes CKT candidate scoring and any paired adaptation trials. ThoughtVLA achieves the highest FSR and lowest forgetting among the listed methods while adding 4.8M/4.9M parameters per stage and retaining 53.0/53.5 MiB on LIBERO-Object/RoboTwin 2.0. Both are the lowest costs in their respective columns, consistent with learning action-head LoRA updates and retaining task knowledge and references for skill reasoning and transfer. Training takes 2.3/6.5 hours per incremental stage, second only to SkillsCrafter, while throughput reaches 32.4/20.8 samples/s and post-selection latency falls to 380.0/600.0 ms, the best values in both settings. These results pair continual performance with compact retained state and low action-prediction cost after skill selection.

Table A11: **Continual performance and resource cost.** Params denotes mean parameter growth per incremental stage (M), Storage the final persistent state beyond the shared backbone (MiB), Train mean wall-clock training time per incremental stage (h), Samp./s throughput, and Latency batch-1 post-selection action-chunk prediction time at the final stage (ms). Timings use two A100 GPUs. Rates use a 0–100 scale. Ranking marks follow Table 1.

(a) LIBERO-Object

Method	FSR ↑	Forg. ↓	Params ↓	Storage ↓	Train ↓	Samp./s ↑	Latency ↓
ProgLoRA (ACL Findings'25)	51.2	32.4	38.1	872.5	2.4	25.1	500.0
CL-MoE (CVPR'25)	60.4	20.3	38.1	872.5	5.7	16.5	420.0
CLARE (RA-L'26)	68.8	18.4	20.1	394.8	2.7	24.4	450.0
MLR-IFA (CVPR'26)	84.6	16.9	38.1	2914.3	3.1	22.7	420.0
SkillsCrafter (AAAI'26)	90.2	9.3	37.9	867.7	**1.7**	24.2	480.0
ThoughtVLA	**98.6**	**0.6**	**4.8**	**53.0**	2.3	**32.4**	**380.0**

(b) RoboTwin 2.0

Method	FSR ↑	Forg. ↓	Params ↓	Storage ↓	Train ↓	Samp./s ↑	Latency ↓
ProgLoRA (ACL Findings'25)	31.4	7.9	38.1	872.5	7.1	18.0	760.0
CL-MoE (CVPR'25)	28.0	25.7	38.1	872.5	14.0	12.0	700.0
CLARE (RA-L'26)	26.4	28.7	22.5	412.0	6.9	15.2	740.0
MLR-IFA (CVPR'26)	31.2	22.2	38.1	4188.9	7.0	13.2	700.0
SkillsCrafter (AAAI'26)	15.8	5.7	37.9	873.5	**5.1**	17.7	780.0
ThoughtVLA	**52.8**	**2.1**	**4.9**	**53.5**	6.5	**20.8**	**600.0**

C.2 Task Order and Training Randomness

Task order changes both the knowledge available to ESR and the previous skills available for CKT transfer. Table A12 separates this effect from training randomness on LIBERO-Goal and LIBERO-Long: order comparisons fix the training seed, while seed comparisons fix the task order. Across the tested orders and seeds, mean FSR differs from the reference run by at most 0.2 percentage points.

Table A12: **Sensitivity to task order and training randomness.** Three task orders use the same training seed, and seeds 42, 43, and 44 use the original task order. O0–O2 and S0–S2 identify the respective runs. O0 and S0 denote the same reference run. Summaries report the mean and sample standard deviation ($n = 3$, denominator $n - 1$) separately for each factor. Rates use a 0–100 scale.

Setting	Factor	Run	FSR ↑	CSR ↑	Forg. ↓
LIBERO-Goal	Task order	O0	95.0	95.5	1.2
		O1	94.8	95.0	1.1
		O2	95.4	95.6	1.2
		Mean ± sd	95.1 ± 0.3	95.4 ± 0.3	1.2 ± 0.1
	Training seed	S0	95.0	95.5	1.2
		S1	95.2	95.5	1.1
		S2	95.4	95.8	1.1
		Mean ± sd	95.2 ± 0.2	95.6 ± 0.2	1.1 ± 0.1
LIBERO-Long	Task order	O0	81.0	83.4	1.4
		O1	81.4	84.0	1.3
		O2	80.8	83.0	1.6
		Mean ± sd	81.1 ± 0.3	83.5 ± 0.5	1.4 ± 0.2
	Training seed	S0	81.0	83.4	1.4
		S1	81.0	83.5	1.5
		S2	80.6	83.1	1.2
		Mean ± sd	80.9 ± 0.2	83.3 ± 0.2	1.4 ± 0.2

C.3 Ablation Controls

Overall comparison. Variants in Table 5 match the backbone, demonstrations, stage order, and final-update schedule. Base uses ProgLoRA (Yu et al., 2025), with its Goal and Long results repeated from Table 1. The SR, ESR, and CKT variants share the same new-update rank. Base + SR replaces the router with skill reasoning based on accumulated task knowledge, without conditional attribute checking or CKT. Base + SR (static) reuses the same trained LoRA updates, execution maps, skills, and reasoning rules as Base + SR, adding new skills at each stage while fixing shared semantic normalization and instruction-matching weights at their stage-1 values. Comparing the two SR rows isolates shared recalibration within the same reasoning mechanism: FSR rises by 4.8 and 5.2 points on Goal and Long when it is updated. Base + ESR additionally enables conditional attribute checking, and ThoughtVLA further enables CKT; Base + ESR versus ThoughtVLA assesses transfer with ESR enabled.

ESR comparison. Table 6 uses the same held-out paraphrases and task-attribute pairs for skills learned by a prespecified stage s_0. At later stages, selectors share the current library, CKT-trained policies, evolving task knowledge, candidate construction, and evaluation inputs. For evaluation input i at stage s_i, $\hat{q}_i$ denotes the selected skill and q_i the skill associated with the intended manipulation task. The indicator in SR acc. is one when the selected skill matches the target skill ($\hat{q}_i = q_i$), and zero otherwise. The operator Avg_i first averages indicators within each task and input group, then equally over tasks, the two input groups, and evaluated stages $s > s_0$. FSR, CSR, and forgetting use closed-loop evaluation of all encountered tasks throughout the full RoboTwin 2.0 stream.

Semantic ($\mathbf{c}_s^0$) selects directly from scores computed with the current-stage shared normalization N_s, without subsequent evidence refinement. It differs from the ProgLoRA Base in Table 5. ESR w/o attributes selects with $\mathbf{c}_s^1$. ESR w/o gating retains $\mathbf{c}_s^1$ but removes the attribute gate's initial-correction prerequisite and residual-ambiguity condition, retaining availability, applicability, and candidate distinguishability requirements. Initial scoring, evidence weights, and scores outside $\mathcal{C}_s$ remain unchanged. Attribute extraction uses only available images and instructions, without oracle task labels. Test inputs do not enter knowledge construction or tuning, and pairs are specified before outcome comparison.

Oracle skill evaluation. At each stage, oracle evaluation supplies the skill associated with the intended manipulation task and invokes its stored skillwise LoRA composition. The policy checkpoint, observations, instruction, action interface, and evaluation initializations match the non-oracle conditions. No policy is retrained. FSR, CSR, and forgetting follow the same definitions, with each

condition using its own post-learning success rates as the forgetting reference. Oracle SR acc. is 100% by construction; Table 6 reports 55.0% FSR, 58.8% CSR, and 0.2 percentage points of forgetting.

CKT comparison. Table 7 compares transfer strategies with ESR enabled throughout. Variants share ESR rules, task knowledge, skill indices, demonstrations, new-update rank, and final-training budget. Per-skill trial branches use the same pre-stage snapshot. Learned updates and execution maps differ by strategy. Transfer variants share the previous-skill candidate pool, stored coefficients, trial initialization, fitting data, and the held-out acceptance rule. Semantic and gradient scores use $\mathcal{B}_q^{\text{write}}$; validation data support confirmation and test rollouts provide independent evaluation. Gradient alignment ranks each candidate's combined update by

$$\mathbf{G}_q^v = \nabla_{\mathbf{W}^v} \mathcal{E}_q^{\text{write}}\big|_{\boldsymbol{\theta}_0}, \qquad A_q^{\text{grad}}(\mathcal{F}) = \sum_v w_v \frac{\langle -\mathbf{G}_q^v, \mathbf{D}_{\mathcal{F}}^v \rangle_F}{\|\mathbf{G}_q^v\|_F \|\mathbf{D}_{\mathcal{F}}^v\|_F}. \tag{A13}$$

Here, $\mathbf{D}_{\mathcal{F}}^v$ is the candidate's combined update and w_v is the layer weight. A zero-norm layer contributes zero without renormalizing the weights; no previous-skill gradients or demonstrations are used. Full-residual ranking uses $-\sum_v w_v \|\mathbf{R}_{q,\mathcal{F}}^v\|_F^2$. The highest-scoring candidate receives at most one paired trial, with no retry after rejection. CKT proposes a candidate only when the maximum $\Delta_r(q, \mathcal{F})$ is positive, so this comparison evaluates complete transfer strategies rather than ranking alone. FSR, CSR, and forgetting measure the full stream.

Overall ablations use LIBERO-Goal and LIBERO-Long; ESR comparisons use RoboTwin 2.0 and CKT comparisons use LIBERO-Long. Ablations use the same task order, training and evaluation seeds, and rollout counts as the full ThoughtVLA evaluation on the corresponding dataset.

Matched action-head-only controls. Table A13 compares ThoughtVLA with SkillsCrafter and DMPEL restricted to action-head LoRA on LIBERO-Object and Goal; the two baselines retain their continual-learning and replay rules. The comparison matches the initial VLA, rank and scaling, demonstrations, task order, optimizer, final-update budget, and evaluation initializations. ThoughtVLA's row repeats Table 1. It exceeds the stronger restricted baseline by 12.2 FSR points on Object and 10.4 on Goal, with lower forgetting on both streams.

Table A13: **Matched action-head-only controls on LIBERO-Object and LIBERO-Goal.** FSR and CSR are percentages; Forg. is in percentage points. Purple bold and blue underlining mark the best and second-best listed values.

	LIBERO-Object			LIBERO-Goal		
Method	FSR ↑	CSR ↑	Forg. ↓	FSR ↑	CSR ↑	Forg. ↓
SkillsCrafter (head only)	81.4	82.6	14.5	<u>84.6</u>	<u>87.3</u>	<u>8.0</u>
DMPEL (head only)	<u>86.4</u>	<u>88.1</u>	<u>12.5</u>	75.0	80.3	13.8
ThoughtVLA	**98.6**	**99.5**	**0.6**	**95.0**	**95.5**	**1.2**

D LIMITATIONS

ThoughtVLA assumes known training-stage boundaries and demonstration groupings, and its retained LoRA updates and task knowledge grow with the skill library. Continual robot learning also encompasses less structured task arrivals, longer streams, and a wider range of embodiments than those studied here. Extending evolving skill reasoning and complementary transfer to these settings is a natural direction for future work. More broadly, understanding how accumulated task knowledge can continue to guide skill use as experience becomes more diverse remains an important question for continual VLA research.