GROW: GROUNDING WITH LONG-TERM AWARENESS FOR SKILL-INCREMENTAL ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Skill-incremental robotic manipulation requires continuously acquiring new skills while preserving previously learned capabilities. A widely adopted paradigm formulates manipulation as keyframe-wise action prediction, where the policy predicts the next keyframe action from the current observation and instruction. Applying skill-incremental learning to such keyframe-based manipulation introduces a fundamental level mismatch: local keyframe supervision and replay provide limited explicit guidance for preserving the long-term semantics of complete skill execution. To address this discrepancy, we propose **GROW** (**Gr**ounding with L**o**ng-Term A**w**areness), a continual manipulation framework that incorporates long-horizon action semantics across perception, action prediction, and replay, enabling skill-level semantics to be better captured and preserved across stages. At the perception level, GROW introduces dual-horizon grounding to regularize cross-modal attention toward both immediate interaction targets and long-horizon task goals. An auxiliary action chain predictor further captures how each skill evolves toward completion through instruction-conditioned future action prediction. The resulting action-chain predictions are additionally exploited for chain-based replay herding, enabling replay selection based on trajectory-level skill semantics. Experiments on RLBench simulation and real-world manipulation tasks show that GROW consistently improves performance over prior skill-incremental learning approaches. Code will be released upon acceptance.

1 INTRODUCTION

Robotic manipulation (Shridhar et al., 2023; Shan et al., 2026) has attracted increasing attention, benefiting substantially from the rapid development of multi-modal perception (Cao et al., 2024; Qian et al., 2026), language-conditioning policy learning (Ze et al., 2023; Zhao et al., 2025), and scalable imitation learning (Kim et al., 2024; Ni et al., 2026). While recent universal policies can handle multiple manipulation tasks (Kim et al., 2024; Hou et al., 2026), most assume a predefined set of tasks or skills of interest during training and deployment. Such a paradigm limits the adaptability of intelligent agents, which should be capable of incrementally acquiring new manipulation skills during interaction with human users while retaining the previously learned behaviors.

To address this gap, incremental learning for robotic manipulation has emerged as a promising solution, with approaches that diverge across their settings and designs. Early attempts, motivated by lifelong-learning benchmarks such as LIBERO (Liu et al., 2023), mainly formulate continual manipulation as sequential adaptation to individual tasks with variations in objects, spatial configurations, task goals, or their compositions. Although such variations may substantially change observations or instructions, many successive tasks share common interaction patterns despite variations. Various learning-without-forgetting mechanisms have been explored, such as segment discovery (Wan et al., 2024), task-specific parameter tuning (Liu et al., 2024), and latent replay (Roy et al., 2025; Yu et al., 2026). More recently, iManip (Zheng et al., 2025) considers a more behaviorally diverse incremental setting named skill-incremental learning (SIL), where successive stages introduce distinct manipulation skills (e.g., "close the jar" vs. "open the drawer") that may require different interaction patterns and action primitives. To accommodate such skill expansion, iManip leverages keyframe-based manipulation policies (Shridhar et al., 2023; Lu et al., 2024) that sparsify dense trajectories into critical keyframe states and formulate manipulation as sequential next-keyframe prediction.

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.

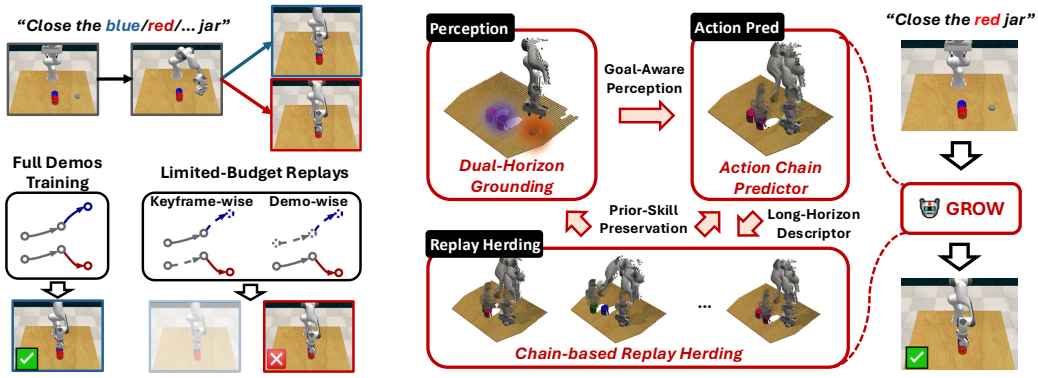

Figure 1: Illustration of the level mismatch in keyframe-based SIL (left) and GROW (right). Taking "close the blue /red jar" as an example, the original next-keyframe prediction may provide limited guidance for preserving its instruction-conditioned execution contexts. GROW is designed to preserve such long-horizon semantics throughout perception, action prediction, and replay herding.

While such a compact formulation is well suited for representing diverse manipulation behaviors, directly extending it to SIL exposes a critical level mismatch: *learning and replay operate on individual keyframes, whereas the knowledge to preserve spans complete skill execution.* Successful manipulation requires preserving how intermediate interactions relate to the instruction-conditioned task goal, whereas keyframe-based SIL optimizes and rehearses these interactions primarily through local next-keyframe prediction. For instance, "close the blue/red jar" may share similar local motions when picking up the lid, while correctly associating these actions with the instructed target requires preserving their long-horizon execution context (see Figure 1). This limitation is less pronounced when complete demonstrations of all skills are jointly available, as the policy is repeatedly exposed to keyframes spanning different stages of each instruction-conditioned execution. Under SIL, however, previous demonstrations are reduced to a limited replay subset. Under a fixed replay budget, keyframe-wise replay (Zheng et al., 2025) provides broader coverage of local states, actions, and instruction variants, but retains only sparse fragments of each skill execution. Conversely, replaying complete demonstrations preserves execution continuity at the cost of substantially reduced sample diversity. This trade-off motivates preserving long-horizon execution semantics without requiring full-trajectory replay.

Motivated by this discrepancy, we argue that SIL should preserve not only local action mappings, but also the long-horizon semantics relating current interactions to subsequent skill progression and the eventual task goal. To this end, we introduce **Gr**ounding with L**o**ng-Term A**w**areness (**GROW**), which augments conventional keyframe-based manipulation with long-horizon semantics to better capture and preserve skill-level execution knowledge across incremental stages. Specifically, GROW incorporates long-term awareness across perception, action prediction, and replay herding. At the perception level, dual-horizon grounding is introduced to regularize the early cross-attention layer towards both immediate interaction regions and long-term task goals. Beyond these isolated spatial anchors, an action chain predictor models the instruction-conditioned future progression of each skill, providing long-horizon execution context for immediate action prediction. The resulting action-chain representations are further exploited for chain-based replay herding, enabling representative replay samples to be selected based on trajectory-level execution patterns instead of local actions. To summarize, our contributions are as follows:

- We identify a critical mismatch between keyframe-level learning and skill-level knowledge preservation in skill-incremental manipulation, motivating long-horizon semantic modeling beyond local action mappings.
- We propose GROW, which introduces long-term awareness into perception, action prediction, and replay herding. Together, it enables long-horizon execution knowledge to be explicitly captured and propagated across incremental stages.
- Extensive experiments on RLBench across three manipulation backbones show that GROW consistently outperforms existing incremental-learning baselines. Real-world experiments further demonstrate its applicability beyond simulation.

2 Related Works

Incremental Learning for Robotic Manipulation. The existing incremental learning approaches for robotic manipulation can be divided into two types based on their problem formulation. Motivated by the preliminary exploration of life-long learning proposed in LIBERO (Liu et al., 2023), the major works define the continual manipulation problem as a sequential adaptation of individual tasks with controllable variations, such as objects, spatial locations, goals, or their combinations. Based on these settings, multiple attempts have been explored mainly based on existing incremental strategies (Gao et al., 2021; Liu et al., 2023; Wan et al., 2024; Zhu et al., 2022; Liu et al., 2024; Lee et al., 2024; Roy et al., 2025; Yu et al., 2026). Another line considers skill-incremental learning (SIL), which is primarily introduced by iManip (Zheng et al., 2025). For SIL, distinct skills are acquired sequentially, and iManip mitigates skill forgetting through temporally balanced keyframe replay and extendable action prompts. Different from these approaches, GROW focuses on the level mismatch inherent to keyframe-based skill-incremental learning, explicitly preserving long-horizon execution semantics across perception, action prediction, and replay.

Perception Grounding for Robotic Manipulation. Task-relevant visual grounding has been widely explored to improve manipulation accuracy, which is usually leveraged with Vision-Language-Action (VLA) models. Some approaches leverage specialized external models, such as RoboGround (Huang et al., 2025) and VIP (Li et al., 2025) leveraging LISA (Lai et al., 2024) and YOLOv11 (Khanam & Hussain, 2024), respectively. Another branch is to take advantage of the powerful visual-language reasoning ability of VLA, Vision-Language Models (VLMs), or Large Language Models (LLMs) (Zawalski et al., 2024; Song et al., 2026). In contrast, GROW utilizes an immediate interaction target and an eventual execution goal inherent in the training data to regularize the internal cross-modal attention without involving external supervision signals or large models. This aims to rectify early perception for both local action prediction and long-horizon skill execution, which is particularly important for knowledge preservation under incremental learning.

Long-Horizon Execution Modeling. Modeling multiple future actions has also proven effective for robotic manipulation. Representative works include ACT (Zhao et al., 2023), which adopts action chunking to achieve affordable smooth actions in fine-grained control, and diffusion-based policies (Chi et al., 2023; Xian et al., 2023; Ke et al., 2024). Other approaches directly synthesize sequences of end-effector waypoints or decompose long-horizon tasks into intermediate interaction steps (Mees et al., 2022; Ni et al., 2024; Jain et al., 2025). These approaches primarily employ hierarchical execution structures for action generation or control. In contrast, GROW retains the next-keyframe prediction policy, leveraging action-chain prediction as an auxiliary supervision signal to encode instruction-conditioned future progression for better skill-level knowledge preservation.

3 Methodology

In this section, we formulate the problem of SIL and then introduce the architecture of our proposed GROW, a PerceiverIO-based (Jaegle et al., 2021) framework for skill-incremental manipulation. GROW aims to preserve the skill-level knowledge by incorporating long-term action awareness across perception, action prediction, and replay, as illustrated in Figure 2. At the perception level, dual-horizon grounding regularizes early cross-modal attention toward task-relevant regions. At the action level, an action chain predictor captures long-horizon execution semantics through multi-step action prediction, whose representations are further exploited for Chain-based Replay Herding. For incremental updates, we adopt stage-wise LoRA with initialization, merging, and calibration, maintaining a fixed model capacity.

Problem Formulation. Following existing protocol (Ze et al., 2023; Lu et al., 2024), we formulate the manipulation task as a keyframe-prediction problem. Specifically, a dataset $\mathcal{D} = \{\zeta_1, \zeta_2, ..., \zeta_n\}$ contains n expert demonstrations, each of which is composed of keyframes when joint velocities approach zero while the gripper's openness remains consistent. Each demonstration $\zeta_i = \{\mathcal{A}_i, \mathcal{O}_i, \mathcal{P}_i, \mathbf{l}_i\}$ contains a sequence of actions $\mathcal{A}_i = \{\mathbf{a}_t\}_{t=1}^{T_i}$, observations $\mathcal{O}_i = \{\mathbf{o}_t\}_{t=1}^{T_i}$, proprioception states $\mathcal{P}_i = \{\mathbf{p}_t\}_{t=1}^{T_i}$, and the linguistic instruction $\mathbf{l}_i$ of the i-th demonstration, respectively. $\mathbf{a}_t = \{\mathbf{a}_t^{\text{pos}}, \mathbf{a}_t^{\text{rot}}, a_t^{\text{open}}, a_t^{\text{col}}\}$ denotes an action associated with keyframe t, including the gripper position $\mathbf{a}_t^{\text{pos}}$ and rotation $\mathbf{a}_t^{\text{rot}}$, gripper openness $a_t^{\text{open}} \in [0, 1]$, and collision avoidance $a_t^{\text{col}} \in [0, 1]$, while an observation $\mathbf{o}_t \in \mathbb{R}^{H \times W \times L \times 3}$ is captured as the colorful 3D voxels from an

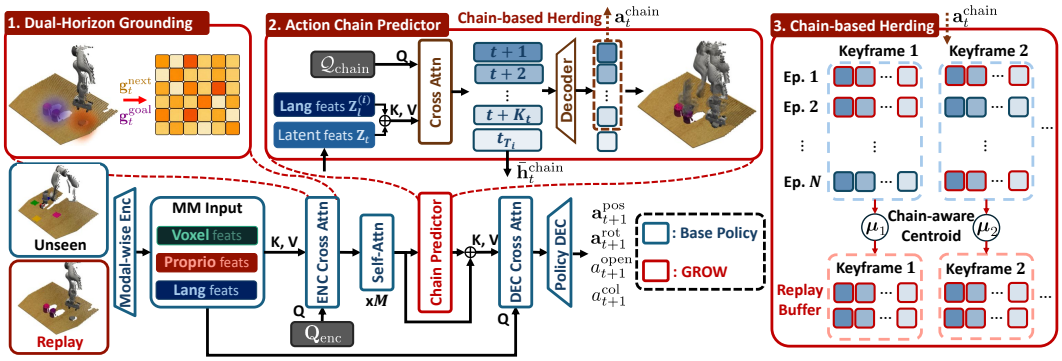

Figure 2: Overall structure of GROW.

RGB-D image. $\mathbf{p}_t \in \mathbb{R}^4$ denotes the gripper state composed of the finger positions and openness. The model $\phi(\cdot)$ is trained to predict actions $\hat{\mathbf{a}}_{t+1} = \phi(\mathbf{o}_t, \mathbf{p}_t, \mathbf{l}_i)$.

Under the skill-incremental setting (Zheng et al., 2025), learning proceeds over a sequence of stages $s = 1, \ldots, S$, where each stage introduces an unseen dataset $\mathcal{D}_{\text{new}}^{(s)}$ corresponding to some novel manipulation skills with interaction semantics distinct from the previously learned skills (e.g., stacking blocks vs. closing a jar). Starting from the model learned at $s-1$, the objective is to obtain a model $\phi^{(s)}$ that acquires the new skills while retaining competence on all the previously learned skills.

3.1 Keyframe-based PerceiverIO Backbones

Existing keyframe-based manipulation approaches, such as PerACT (Shridhar et al., 2023), GNFactor (Ze et al., 2023), and ManiGaussian (Lu et al., 2024), commonly adopt PerceiverIO-style architectures to aggregate multi-modal observations and predict subsequent actions. At keyframe t, observation, linguistic, and proprioceptive features are first flattened, encoded, and concatenated into a multi-modal representation $\mathbf{X}_t$. The Perceiver encoder employs a set of latent queries $\mathbf{Q}_{\text{enc}}$ to aggregate multi-modal information through cross attention (Vaswani et al., 2017), denoted as:

$$\mathbf{Z}_t^0 = \texttt{CrossAttn}_{\text{enc}}(\mathbf{Q}_{\text{enc}}, \mathbf{X}_t, \mathbf{X}_t), \tag{1}$$

where $\mathbf{Q}_{\text{enc}}$ is the query input, while $\mathbf{X}_t$ serves as the keys and values input; all are encoded with input-specific linear weights before the attention computation. The extracted latent features $\mathbf{Z}_t^0$ are then aggregated by several self-attention blocks, whose output is queried by the multi-modal representation $\mathbf{X}_t$, formulated as:

$$\mathbf{Z}_t = \texttt{SelfAttn}_{\times M}(\mathbf{Z}_t^0), \quad \mathbf{H}_t = \texttt{CrossAttn}_{\text{dec}}(\mathbf{X}_t, \mathbf{Z}_t, \mathbf{Z}_t), \tag{2}$$

where $\mathbf{Z}_t$ denotes the last latent representation before the cross-attention decoder. $\mathbf{H}_t$ is subsequently leveraged to decode action $\hat{\mathbf{a}}_{t+1}$ through the action decoder.

Discussion. Despite the architectural difference, existing PerceiverIO-based methods optimize each observation primarily for next-keyframe prediction. As in Figure 1, such local supervision and keyframe-level replay (Zheng et al., 2025) provide limited explicit guidance on how long-horizon dependencies connect individual interactions into coherent skill execution, motivating GROW.

3.2 GROW: Grounding with Long-Term Awareness

To address this limitation and enable long-horizon semantic propagation across incremental stages, we introduce GROW, which augments conventional keyframe-wise learning with long-term action awareness, allowing skill-level semantics to be explicitly maintained across perception, action prediction, and replay.

Dual-Horizon Grounding. Perception is an important preliminary to provide informative clues for long-horizon skill execution. However, we observe (Figure 4) that the encoder cross-attention of existing PerceiverIO-based policies can remain diffuse over task-irrelevant regions, and such grounding becomes increasingly fragile across incremental stages. To reinforce the connection between

perception and skill execution, GROW regularizes the encoder cross-attention through two complementary spatial anchors, including the next-keyframe position for immediate interaction and the terminal-keyframe position for long-horizon task context. Specifically, given the input multi-modal representation $\mathbf{X}_t$, the attention map of $\texttt{CrossAttn}_{\text{enc}}$ in Equation 1 is preserved as:

$$\mathbf{A}_t = \text{Softmax}(\mathbf{Q}_{\text{enc}}\mathbf{W}_Q(\mathbf{X}_t\mathbf{W}_K)^\top/\sqrt{d}), \tag{3}$$

where $\mathbf{W}_Q$ and $\mathbf{W}_K$ denote the learnable query and key projections of the encoder cross-attention. Given $\mathbf{A}_t$ as the attention map between $\mathbf{Q}_{\text{enc}}$ and $\mathbf{X}_t$, since $\mathbf{X}_t$ contains both spatial observations and non-spatial modalities, we retain only the attention scores correlated to the voxelized observation $\mathbf{o}_t$ through mask selection, aggregate them across latent queries and attention heads, and normalize the results to form an attention distribution α_t as:

$$\alpha_t = \text{Norm}(\text{Agg}(\mathbf{A}_t[:, \mathcal{I}_{\text{voxel}}])), \tag{4}$$

where $\mathcal{I}_{\text{voxel}}$ denotes the indices of voxel tokens in $\mathbf{X}_t$ and $\text{Norm}(\cdot)$ indicates the normalization across all selected elements. The next-step and goal grounding anchors are then formulated as 3D Gaussian distributions centered at the spatial location related to the immediate action $\mathbf{a}_{t+1}$ and goal action $\mathbf{a}_{T_i}$, formulated as:

$$\mathbf{g}_t^{\text{next}} = \text{Norm}(\mathcal{N}(\mathbf{a}_{t+1}^{\text{pos}}, \sigma^2\mathbf{I})), \quad \mathbf{g}_t^{\text{goal}} = \text{Norm}(\mathcal{N}(\mathbf{a}_{T_i}^{\text{pos}}, \sigma^2\mathbf{I})), \tag{5}$$

where σ denotes the predefined variance of Gaussian distributions for both next-step and goal grounding anchors. The dual-horizon grounding objective is then formulated as:

$$\mathcal{L}_{\text{dh}} = \lambda_{\text{n}} D_{\text{KL}}(\mathbf{g}_t^{\text{next}} \| \alpha_t) + \lambda_{\text{g}} D_{\text{KL}}(\mathbf{g}_t^{\text{goal}} \| \alpha_t), \tag{6}$$

where λ_{n} and λ_{g} are the loss coefficients for immediate and long-horizon grounding, respectively. The former, next-keyframe grounding, encourages the model to focus on the immediate interaction region based on which to predict the next-step action, while the latter goal grounding supplies the long-horizon goal of where the current action ultimately leads, complementing long-horizon context in the early perception stage.

Action Chain Predictor. Dual-horizon grounding preserves semantic cues about both the immediate interaction and terminal goal, but they remain isolated spatial targets, lacking explicit modeling of how execution evolves between them. We therefore introduce an auxiliary action chain predictor that encodes future progression of each skill.

Such future progression is highly instruction-conditioned (Figure 1). Although the instruction $\mathbf{l}_i$ has been incorporated during the early stage of the Perceiver representation, we explicitly reintroduce its representation to provide direct instruction-specific conditioning for long-horizon action chain prediction. Given the latent features extracted by the last self-attention layers $\mathbf{Z}_t$ as in Eq. 2, the action chain predictor first concatenates them with the re-projected linguistic representation as:

$$\mathbf{Z}_{\text{l}}^{(i)} = \mathbf{E}_{\text{l}}^{(i)}\mathbf{W}_{\text{l}}, \quad \bar{\mathbf{Z}}_t = \text{MeanPool}(\mathbf{Z}_t), \quad \tilde{\mathbf{Z}}_t = \mathbf{Z}_{\text{l}}^{(i)} \oplus \bar{\mathbf{Z}}_t, \tag{7}$$

where $\mathbf{E}_{\text{l}}^{(i)}$ denotes the previously encoded linguistic representation existing in $\mathbf{X}_t$ and MeanPool indicates the average pooling along the token dimension. $\mathbf{W}_{\text{l}}$ represents the re-projection in the action chain predictor. The resulting $\tilde{\mathbf{Z}}_t$ subsequently serves as the keys and values for a set of k learnable chain queries $\mathcal{Q}_{\text{chain}} = \{\mathbf{q}_j\}_{j=1}^k$ to extract the action chain representation $\mathbf{H}_t^{\text{chain}}$ as:

$$\mathbf{H}_t^{\text{chain}} = \texttt{SelfAttn}_{\text{chain}}(\texttt{CrossAttn}_{\text{chain}}(\mathcal{Q}_{\text{chain}}, \tilde{\mathbf{Z}}_t, \tilde{\mathbf{Z}}_t)). \tag{8}$$

The valid prediction steps depend on the remaining steps in the trajectory. Specifically, given $R_t = T_i - t$ denoting the number of remaining keyframes, we define $K_t = \min(R_t - 1, k - 1)$ to retrieve the first $K_t + 1$ representations in $\mathbf{H}_t^{\text{chain}}$ and predict the corresponding positions using an MLP-based action decoder. The first K_t representations predict positions of consecutive future keyframes, while the $(K_t + 1)$-th representation predicts the position of the goal, denoted as:

$$\mathcal{A}_t^{\text{chain}} = \{\hat{\mathbf{a}}_{t+1}^{\text{pos}}, \hat{\mathbf{a}}_{t+2}^{\text{pos}}, ...\hat{\mathbf{a}}_{t+K_t}^{\text{pos}}, \hat{\mathbf{a}}_{T_i}^{\text{pos}}\}. \tag{9}$$

Given the action chain representation and predictions, an exponentially decaying weight $w_j^{\text{pred}} = \tau^{j-1}/\sum_{r=1}^{K_t+1}\tau^{r-1}$ controlled by the decay rate $\tau \in (0, 1]$ prioritizes the near-future execution while retaining long-horizon task information. The chain predictor is trained with the objective as:

$$\mathcal{L}_{\text{chain}} = \sum_{j=1}^{K_t} w_j^{\text{pred}} \mathcal{L}_{\text{pos}}\left(\hat{\mathbf{a}}_{t+j}^{\text{pos}}, \mathbf{a}_{t+j}^{\text{pos}}\right) + w_{K_t+1}^{\text{pred}} \mathcal{L}_{\text{pos}}\left(\hat{\mathbf{a}}_{T_i}^{\text{pos}}, \mathbf{a}_{T_i}^{\text{pos}}\right), \tag{10}$$

where $\mathcal{L}_{\text{pos}}$ denotes the identical pose loss function used for the next-keyframe prediction. $\mathbf{H}_t^{\text{chain}}$ is then detached and injected into the backbone to provide long-term awareness as:

$$\bar{\mathbf{h}}_t^{\text{chain}} = \sum_{j=1}^{k} w_j^{\text{feat}} \mathbf{h}_{t,j}^{\text{chain}}, \quad \mathbf{Z}_t^{+} = \texttt{concat}(\mathbf{Z}_t, \bar{\mathbf{h}}_t^{\text{chain}}), \tag{11}$$

where $w_j^{\text{feat}} = \tau^{j-1} / \sum_{r=1}^{k} \tau^{r-1}$ is the feature decaying weight controlled by the same decay factor τ. $\mathbf{h}_{t,j}^{\text{chain}}$ denotes the corresponding j-th chain representation in $\mathbf{H}_t^{\text{chain}}$, while $\mathbf{Z}_t^{+}$ is the long-term aware representation that replaces $\mathbf{Z}_t$ during action decoding in Eq. 2. While only the first $K_t + 1$ query outputs receive direct supervision for a given keyframe, all k queries are shared across training samples and are supervised whenever their corresponding slots become valid. We therefore aggregate the complete query representation as a fixed-dimensional long-horizon context for immediate action prediction.

Chain-based Replay Herding. Experience replay is widely adopted in incremental learning to mitigate catastrophic forgetting, where a limited subset of samples from previous stages is retained and replayed during subsequent training. Under keyframe-based manipulation, existing approaches (Zheng et al., 2025) propose to utilize per-keyframe prediction statistics or features as the selection criteria. Despite being consistent with the next-keyframe prediction paradigm, such strategies categorize each keyframe sample largely based on its immediate movement, providing limited information about how the remaining manipulation evolves for task completeness. In contrast, GROW implements a chain-based replay herding that leverages the long-horizon action information from the chain predictor to achieve replay herding with long-term awareness. Taking a keyframe sample t as an example, given the predicted chain logits $\mathcal{A}_t^{\text{chain}}$, we compute its corresponding long-term prediction descriptors through pooling using the same w_j^{pred}, formulated as:

$$\mathbf{a}_t^{\text{chain}} = \sum_{j=1}^{K_t} w_j^{\text{pred}} \hat{\mathbf{a}}_{t+j}^{\text{pos}} + w_{K_t+1}^{\text{pred}} \hat{\mathbf{a}}_{T_i}^{\text{pos}}. \tag{12}$$

For replay herding, we conduct sample selection independently within each previously learned task. Let $\tilde{\mathcal{D}} = \{\zeta_i\}_{i=1}^{N}$ denote the demonstration subset belonging to the current task, and M denote the representative trajectory length computed as the average keyframe number of ζ_i. For each keyframe position $m \in \{1, \ldots, M\}$, we compute the corresponding chain-aware centroid by averaging the long-term prediction descriptors of these keyframes:

$$\boldsymbol{\mu}_m = \frac{1}{|\mathcal{I}_m|} \sum_{i \in \mathcal{I}_m} \mathbf{a}_{i,m}^{\text{chain}}, \tag{13}$$

where $\mathcal{I}_m$ contains the indices whose corresponding demonstrations contains at least m keyframes. $\mathbf{a}_{i,m}^{\text{chain}}$ denotes the pooled action-chain logit of the m-th keyframe from demonstration ζ_i as in Eq. 12. Following mean-based herding (Rebuffi et al., 2017; Castro et al., 2018), candidate samples are ranked according to their Euclidean distance to the corresponding centroid $d_{i,m} = \left\| \mathbf{a}_{i,m}^{\text{chain}} - \boldsymbol{\mu}_m \right\|_2$. Given a budget of B_m samples at position m, we retain the candidates with the smallest distances:

$$\mathcal{R}_m = \underset{\mathcal{S} \subseteq \mathcal{I}_m}{\arg\min} \sum_{i \in \mathcal{S}} d_{i,m}, \quad \text{s.t. } |\mathcal{S}| = B_m. \tag{14}$$

The whole replay indices for the task are therefore represented as $\mathcal{R} = \bigcup_{m=1}^{M} \{(m, R_m)\}$. For each replay index, we store the corresponding keyframe input, its next-step ground-truth action, and the valid future ground-truth action positions required to supervise the action chain predictor. These replay samples are then stochastically mixed with new-task data during incremental training.

3.3 Incremental Update and Objectives.

For each incremental stage, we employ a LoRA-based (Hu et al., 2021) update strategy to update the shared backbone with a constrained number of trainable parameters. To maintain a constant adaptation budget while preserving sufficient plasticity for newly introduced skills, GROW further adopts stage-wise LoRA initialization, merging, and calibration. Specifically, a fresh set of LoRA parameters is initialized at the beginning of each stage and optimized along with the chain predictor

and the PerceiverIO's action decoder, while the other components in PerceiverIO remain frozen. After the training process, the LoRA components are merged back into the backbone through summation ($\mathbf{W} = \mathbf{W}_0 + \mathbf{B}\mathbf{A}\alpha/r$ as in (Hu et al., 2021)), followed by a short calibration fine-tuning using the same objectives and training data but a reduced learning rate. Details about LoRA-targeted components and implementations are included in our appendix.

The overall learning objectives consist of four components. First, a conventional action loss is adopted to facilitate the imitation learning of the backbone, formulated as:

$$\mathcal{L}_{\text{action}} = \mathcal{L}_{\text{pos}}(\hat{\mathbf{a}}^{\text{pos}}_{t+1}, \mathbf{a}^{\text{pos}}_{t+1}) + \mathcal{L}_{\text{rot}}(\hat{\mathbf{a}}^{\text{rot}}_{t+1}, \mathbf{a}^{\text{rot}}_{t+1}) + \mathcal{L}_{\text{open}}(\hat{a}^{\text{open}}_{t+1}, a^{\text{open}}_{t+1}) + \mathcal{L}_{\text{col}}(\hat{a}^{\text{col}}_{t+1}, a^{\text{col}}_{t+1}), \quad (15)$$

where $\hat{\mathbf{a}}^{\text{pos}}_{t+1}, \hat{\mathbf{a}}^{\text{rot}}_{t+1}, \hat{a}^{\text{open}}_{t+1}, \hat{a}^{\text{col}}_{t+1}$ denote the backbone's next-keyframe predictions of positions, rotations, collision, and openness, respectively. In practice, the loss function for each component is implemented as cross-entropy loss following the protocol of previous works (Lu et al., 2024; Zheng et al., 2025). Additionally, a knowledge distillation loss is applied to the replay samples to better preserve previously acquired knowledge using the model from the previous stage as the teacher:

$$\mathcal{L}_{\text{dis}} = \text{MSE}(\hat{\mathbf{a}}^{\text{pos}}_{t+1}, \tilde{\mathbf{a}}^{\text{pos}}_{t+1}) + \text{MSE}(\hat{\mathbf{a}}^{\text{rot}}_{t+1}, \tilde{\mathbf{a}}^{\text{rot}}_{t+1}) + \text{MSE}(\hat{a}^{\text{open}}_{t+1}, \tilde{a}^{\text{open}}_{t+1}) + \text{MSE}(\hat{a}^{\text{col}}_{t+1}, \tilde{a}^{\text{col}}_{t+1}), \quad (16)$$

where $\tilde{\mathbf{a}}^{\text{pos}}_{t+1}, \tilde{\mathbf{a}}^{\text{rot}}_{t+1}, \tilde{a}^{\text{open}}_{t+1}, \tilde{a}^{\text{col}}_{t+1}$ denote the corresponding predictions from the trained model of the previous stage. With the GROW-specific objectives, the total loss function can be represented as:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{action}} + \lambda_{\text{dis}}\mathcal{L}_{\text{dis}} + \lambda_{\text{chain}}\mathcal{L}_{\text{chain}} + \mathcal{L}_{\text{dh}}, \quad (17)$$

where $\lambda_{\text{dis}}, \lambda_{\text{chain}}$ are the coefficients for the distillation loss and chain prediction loss, respectively.

4 Experimental Results

Benchmark Details. Following existing works (Lu et al., 2024; Zheng et al., 2025), our experiments are conducted on 10 manipulation skills in RLBench (James et al., 2020), supplemented with real-world rollouts involving 3 skills. For simulation, each skill contains 20 demonstrations sampled from at least 2 variations, while the evaluation involves 25 episodes per task. Following the SIL protocol of iManip, we utilize the same configuration notation "B*n*-*k*N*m*", which indicates agents are pre-trained on *n* base skills and incrementally updated across *k* stages, each of which introduces *m* unseen skills. For real-world rollouts, we collect 20 demonstration episodes as training data and evaluate each method with 10 rollouts. More details are included in our appendix.

Baseline Details. GROW is mainly compared with **iManip** (Zheng et al., 2025), the existing state-of-the-art (SOTA) keyframe-based SIL method, and the baseline used in their comparison called **TIB** (Rebuffi et al., 2017). We also adopt 3 baselines mostly adopted in lifelong learning (Liu et al., 2023), including sequential fine-tuning (**Seq-FT**), fine-tuning with a Fisher-information-based regularizer (**EWC**) (Kirkpatrick et al., 2017), and **TAIL** (Liu et al., 2024; Yu et al., 2026). Our main comparison is conducted across 3 keyframe-based backbones, including PerAct (Shridhar et al., 2023), GNFactor (Ze et al., 2023), and ManiGaussian (Lu et al., 2024). The other incremental settings and ablation studies in simulation are based on ManiGaussian, while the real-world rollouts are all based on PerAct. *Note that we utilize our self-implemented version of iManip and TIB, as their code is not available till submission*. More details of baselines are in our appendix.

Implementation Details and Metrics. All experiments are conducted on an NVIDIA RTX PRO 6000 GPU using 100k iterations for base pre-training and 50k iterations for each incremental stage. Most input preprocessing protocols follow existing works (Shridhar et al., 2023; Ze et al., 2023; Lu et al., 2024). For each incremental stage, we compute the success rate of tasks from previous stages (Old), tasks newly introduced (New), and overall average (All). More details are in our appendix.

4.1 Main Comparison with SOTA Methods

As shown in Table 1, we compared GROW under the B5-5N1 setting using three different backbones, including PA, GF, and MG. Compared with baseline approaches, GROW consistently achieves SOTA performance across all backbones in terms of average overall performance. Specifically with MG, it outperforms iManip across old skills preservation, new skill comprehension, and overall performance, with a noticeable gap of 11.7%, 3.2%, and 10.7%, respectively. Similar improvement can be observed on GF and PA, where GROW surpasses iManip by 3.8% and 3.5% on

Table 1: Stage-wise performance comparison under B5-5N1. The first column indicates the backbone (BB) used for each row, including PerAct (PA), GNFactor (GF), and ManiGaussian (MG).

BB	Method	Base	Stage 1			Stage 2			Stage 3			Stage 4			Stage 5			Avg		
			Old	New	All	Old	New	All	Old	New	All	Old	New	All	Old	New	All	Old	New	All
PA	Seq-FT	**34.4**	10.4	56.0	18.0	5.3	8.0	5.7	4.6	16.0	6.0	1.0	**28.0**	4.0	3.6	**24.0**	5.6	5.0	**26.4**	7.9
	EWC	34.4	14.4	**68.0**	23.3	6.0	0.0	5.1	4.0	4.0	4.0	0.5	8.0	1.3	2.7	0.0	2.4	5.5	16.0	7.2
	TAIL	**34.4**	25.6	12.0	23.3	18.7	1.3	16.2	12.9	3.0	11.6	11.6	4.8	10.8	8.8	0.0	7.9	15.5	4.2	14.0
	TIB	**34.4**	28.0	36.0	29.3	16.0	0.0	13.7	**24.6**	16.0	**23.5**	14.0	8.0	13.3	15.6	0.0	14.0	19.6	12.0	18.8
	iManip	**34.4**	24.0	16.0	22.7	22.7	12.0	21.1	17.1	4.0	15.5	**19.5**	12.0	**18.7**	**18.7**	0.0	**16.8**	20.4	8.0	19.0
	GROW	33.6	**34.4**	20.0	**32.0**	**29.3**	**16.0**	**27.4**	21.1	**20.0**	21.0	18.0	8.0	16.9	16.9	0.0	15.2	**23.9**	12.8	**22.5**
GF	Seq-FT	48.0	0.0	**84.0**	14.0	6.7	8.0	6.9	1.1	**68.0**	9.5	0.0	12.0	1.3	2.2	**12.0**	3.2	2.0	**36.8**	7.0
	EWC	48.0	8.0	56.0	16.0	13.3	16.0	13.7	1.1	44.0	6.5	1.5	4.0	1.8	2.7	0.0	2.4	5.3	24.0	8.1
	TAIL	48.0	29.6	10.0	26.3	20.7	6.7	18.7	16.1	6.0	14.9	12.0	3.2	11.0	11.7	0.0	10.5	18.0	5.2	16.3
	TIB	48.0	17.6	24.0	18.7	16.0	**16.0**	16.0	**20.0**	56.0	**24.5**	19.5	**32.0**	20.9	13.3	0.0	12.0	17.3	25.6	18.4
	iManip	48.0	32.8	28.0	32.0	20.7	0.0	17.7	17.7	24.0	18.5	21.0	16.0	20.4	18.7	0.0	16.8	22.2	13.6	21.1
	GROW	**49.6**	**38.4**	32.0	**37.3**	**31.3**	8.0	**28.0**	16.0	24.0	17.0	**21.5**	24.0	**21.7**	**22.7**	0.0	**20.4**	**26.0**	17.6	**24.9**
MG	Seq-FT	**60.0**	12.8	**72.0**	22.7	3.3	**40.0**	8.6	4.0	**64.0**	11.5	2.5	12.0	3.6	2.7	**12.0**	3.6	5.1	**40.0**	10.0
	EWC	**60.0**	8.8	28.0	12.0	11.3	8.0	10.9	1.7	52.0	8.0	5.5	4.0	5.3	4.0	0.0	3.6	6.3	18.4	8.0
	TAIL	**60.0**	34.0	16.0	31.0	26.0	8.0	23.4	16.6	10.0	15.8	14.5	13.6	14.4	10.8	0.7	9.8	20.4	9.7	18.9
	TIB	**60.0**	16.0	28.0	18.0	28.7	0.0	24.6	19.4	56.0	24.0	18.0	8.0	16.9	25.3	0.0	22.8	21.5	18.4	21.3
	iManip	**60.0**	33.6	28.0	32.7	24.0	16.0	22.9	16.0	24.0	17.0	22.0	12.0	20.9	16.0	0.0	14.4	22.3	16.0	21.6
	GROW	59.2	**40.0**	36.0	**39.3**	**41.3**	16.0	**37.7**	**25.1**	28.0	**25.5**	**33.5**	**16.0**	**31.6**	**30.2**	0.0	**27.2**	**34.0**	19.2	**32.3**

Table 2: Ablation studies under B5-5N1. Here, figures denote performance averaged over all incremental stages for compactness. Stage-wise performance is in our appendix.

ID	Ablation	Incremental Only				Base + Incremental			
		Base	Old	New	All	Base	Old	New	All
1	w/o next grounding	59.2	27.9	13.6	26.1	47.2	19.0	**29.6**	20.4
2	w/o goal grounding	59.2	32.9	20.8	31.5	52.0	22.1	13.6	21.1
3	w/o action chain prediction	59.2	27.0	**24.0**	26.6	53.0	24.4	11.2	22.7
4	Chain herding vs. TIB herding	59.2	29.3	13.6	27.3	-	-	-	-
5	w/o chain feat injection	59.2	31.3	15.2	29.4	54.4	24.5	18.4	23.8
6	w/o chain decay weight	59.2	26.8	16.8	25.4	55.3	25.0	15.2	23.8
7	w/o lang token in	59.2	24.3	15.2	23.2	48.8	28.7	9.6	26.1
-	iManip	**60.0**	22.3	16.0	21.6	**60.0**	22.3	16.0	21.6
-	GROW	59.2	**34.0**	19.2	**32.3**	59.2	**34.0**	19.2	**32.3**

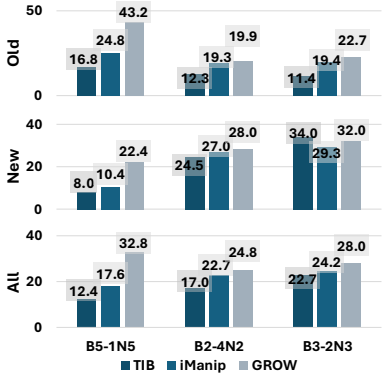

Figure 3: Performance of B5-1N5, B2-2N4, and B3-2N3.

overall performance, respectively. For new skill comprehension, Seq-FT and TIB achieve superior performance to GROW, mainly because they tune the global parameters of networks, which also causes their more severe degradation in old skill preservation. This demonstrates that GROW achieves better new skill comprehension while alleviating the forgetting of previous skills.

4.2 Ablation Studies

Component-wise Effectiveness. We first validate the contributions of the key components that compose GROW (Table 2) under two scenarios: disabling each component only during the incremental learning (*Incremental Only*), and removing the component from both base pretraining and incremental stages (*Base + Incremental*). Removing either next-keyframe or goal grounding consistently degrades the overall performance, confirming the complementary benefits of immediate interaction and long-horizon goal supervision. Disabling the chain predictor leads to a similar drop, emphasizing the importance of modeling skill progression. Replacing chain-based herding with keyframe-based herding also causes a noticeable drop from 32.3% to 27.3%, illustrating the improvement brought by long-horizon-aware replay selection.

Specific Designs of Action Chain Predictor. We further evaluate some key designs in the action chain predictor (Table 2). Without the aid of chain feature injection, both incremental and full training suffer from a non-trivial performance decrease, indicating the importance of using long-horizon representation for next-step prediction. Disabling the decay weight causes a drop of more than 7% in old skill preservation, suggesting that emphasizing nearer-term progression while retaining long-

Table 3: B5-1N1 performance.

Method	Base	Slide the Block			Put in Drawer			Reach and Drag			Push Buttons			Stack Blocks			Avg		
		Old	New	All	Old	New	All	Old	New	All	Old	New	All	Old	New	All	Old	New	All
TIB	**60.0**	16.0	28.0	18.0	16.8	0.0	14.0	14.4	**52.0**	20.7	17.6	**48.0**	22.7	16.8	4.0	14.7	16.3	**26.4**	18.0
iManip	**60.0**	33.6	28.0	32.7	28.8	**12.0**	26.0	24.8	12.0	22.7	18.4	16.0	18.0	23.2	**8.0**	20.7	25.8	15.2	24.0
GROW	59.2	**40.0**	**36.0**	**39.3**	**37.6**	**12.0**	**33.3**	**43.2**	**52.0**	**44.7**	**44.8**	24.0	**41.3**	**28.0**	**8.0**	**24.7**	**38.7**	**26.4**	**36.5**

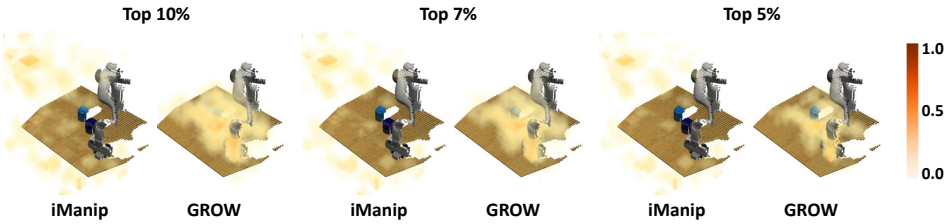

Figure 4: Visualization of attention heatmaps from iManip and GROW (Stage 1 of B5-5N1). Here we visualize the voxel-wise heatmaps with the top-X%, $X \in [10, 7, 5]$ attention scores.

Table 4: Comparison of real-world rollouts.

Manipulation Skill	Base			Stage 1			Stage 2			Avg		
	Seq-FT	iManip	GROW	Seq-FT	iManip	GROW	Seq-FT	iManip	GROW	Seq-FT	iManip	GROW
Put items in baskets	20.0	20.0	**60.0**	10.0	10.0	**40.0**	0.0	10.0	**30.0**	10.0	13.3	**43.3**
Stack blocks	-	-	-	30.0	0.0	**50.0**	0.0	10.0	**30.0**	15.0	5.0	**40.0**
Slide blocks to regions	-	-	-	-	-	-	30.0	30.0	**40.0**	30.0	30.0	**40.0**

horizon context improves old-skill preservation. Removing the linguistic feats from input leads to a similar gap, supporting the importance of explicit instruction conditioning of long-horizon planning.

Performance under Diverse SIL Settings. To rule out the setting bias, we evaluate GROW under 4 more SIL settings along with the competitive TIB and iManip. As in Table 3, the improvement brought by GROW sustains across all short-term SIL with varying new tasks, outperforming existing methods by more than 10% on average. Similar improvements can also be observed on B5-1N5, B2-4N2, and B3-2N3 (Figure 3) across almost all metrics and settings.

Qualitative Comparison. We present the heatmap comparison of the early $\texttt{CrossAttn}_{\text{enc}}$ between iManip and GROW in Figure 4. By incorporating the dual-horizon grounding, GROW can achieve much better concentration during the early perception stage, with a main focus on the task-relevant regions (i.e., jars, the gripper, and workspace).

4.3 Real-World Rollouts

For the real-world experiments, we compare GROW with Seq-FT and iManip using the backbone PerAct (Shridhar et al., 2023) under a SIL setting of B1-2N1. Throughout the SIL process, GROW achieves the highest success rate across all manipulation tasks compared with Seq-FT and iManip. Note that GROW also outperforms the baselines in the base task "Put items in baskets", mainly because our task setting involves localizing and picking small objects (e.g., a strawberry), where the perception grounding of GROW could help substantially. Details are elaborated in our appendix.

5 Conclusion

In this work, we study skill-incremental robotics manipulation from the perspective of the inherent level mismatch between local keyframe-wise learning and long-horizon semantics required to preserve the skill-level knowledge. To address this gap, we propose GROW, which incorporates long-term awareness throughout perception, action prediction, and replay-herding. Extensive experiments conducted on RLBench demonstrate the consistent improvement across different incremental settings and backbones, while ablation studies further justify the effectiveness of the proposed design. The supplemented real-world rollouts further demonstrate the applicability of GROW in real-life scenarios. Our results highlight the significance of long-horizon semantic preservation for continual robot learning, paving the road for future extensions toward more complex skill compositions and long-term real-world adaptation.

AI USE STATEMENT

In this work, we used generative AI tools for implementing methods and designing or providing feedback on research methodology or experiments. We have not used generative AI tools for generating synthetic data sets, proposing or refining hypotheses, assisting with translation, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, or interpreting results, and the rest of the required disclosure tasks do not apply to this work. Additionally, we used generative AI tools for creating or editing software code and editing a research paper to improve readability. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness through multiple steps of verification with intermediate results and inspection of the authors. The final manuscript is proofread by the authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Code of algorithms, simulation benchmarking, and real-world rollouts are included in our supplementary materials and will be released upon acceptance. Details of simulation implementation and real-world rollout protocols are thoroughly introduced in the appendix. For any datasets used in the experiments, a complete description of the data processing is provided in the appendix.

REFERENCES

Haozhi Cao, Yuecong Xu, Jianfei Yang, Pengyu Yin, Xingyu Ji, Shenghai Yuan, and Lihua Xie. Reliable spatial-temporal voxels for multi-modal test-time adaptation. In *European Conference on Computer Vision*, pp. 232–249. Springer, 2024.

Francisco M Castro, Manuel J Marín-Jiménez, Nicolás Guil, Cordelia Schmid, and Karteek Alahari. End-to-end incremental learning. In *European conference on computer vision*, pp. 241–257. Springer, 2018.

Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin CM Burchfiel, and Shuran Song. Diffusion Policy: Visuomotor Policy Learning via Action Diffusion. In *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, July 2023. doi: 10.15607/RSS.2023.XIX.026.

Chongkai Gao, Haichuan Gao, Shangqi Guo, Tianren Zhang, and Feng Chen. Cril: Continual robot imitation learning via generative and prediction model. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 6747–6754. IEEE, 2021.

Bohan Hou, Gen Li, Jindou Jia, Tuo An, Xinying Guo, Sicong Leng, Haoran Geng, Yanjie Ze, Tatsuya Harada, Philip Torr, et al. World model for robot learning: A comprehensive survey. *arXiv preprint arXiv:2605.00080*, 2026.

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

Haifeng Huang, Xinyi Chen, Yilun Chen, Hao Li, Xiaoshen Han, Zehan Wang, Tai Wang, Jiangmiao Pang, and Zhou Zhao. Roboground: Robotic manipulation with grounded vision-language priors. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22540–22550. IEEE, 2025.

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*, pp. 4651–4664. PMLR, 2021.

Shivam Jain, Sachit Sachdeva, and Rohan Paul. Enabling long (er) horizon imitation for manipulation tasks by modeling subgoal transitions. In *9th Annual Conference on Robot Learning*, 2025.

Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J Davison. Rlbench: The robot learning benchmark & learning environment. *IEEE Robotics and Automation Letters*, 5(2):3019–3026, 2020.

Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. *arXiv preprint arXiv:2402.10885*, 2024.

Rahima Khanam and Muhammad Hussain. Yolov11: An overview of the key architectural enhancements. *arXiv preprint arXiv:2410.17725*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

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

Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9579–9589. IEEE, 2024.

Daehee Lee, Minjong Yoo, Woo Kyung Kim, Wonje Choi, and Honguk Woo. Incremental learning of retrievable skills for efficient continual task adaptation. *Advances in neural information processing systems*, 37:17286–17312, 2024.

Zhuoling Li, Liangliang Ren, Jinrong Yang, Yong Zhao, Xiaoyang Wu, Zhenhua Xu, Xiang Bai, and Hengshuang Zhao. Vip: vision instructed pre-training for robotic manipulation. In *Proceedings of the 42nd International Conference on Machine Learning*, ICML'25. JMLR.org, 2025.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36:44776–44791, 2023.

Zuxin Liu, Jesse Zhang, Kavosh Asadi, Yao Liu, Ding Zhao, Shoham Sabach, and Rasool Fakoor. Tail: Task-specific adapters for imitation learning with large pretrained models. In *International Conference on Learning Representations*, volume 2024, pp. 16330–16353, 2024.

Guanxing Lu, Shiyi Zhang, Ziwei Wang, Changliu Liu, Jiwen Lu, and Yansong Tang. Manigaussian: Dynamic gaussian splatting for multi-task robotic manipulation. In *European Conference on Computer Vision*, pp. 349–366. Springer, 2024.

Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data. *IEEE Robotics and Automation Letters*, 7(4):11205–11212, 2022.

Chaojun Ni, Cheng Chen, Xiaofeng Wang, Zheng Zhu, Wenzhao Zheng, Boyuan Wang, Tianrun Chen, Guosheng Zhao, Haoyun Li, Zhehao Dong, Qiang Zhang, Yun Ye, Yang Wang, Guan Huang, and Wenjun Mei. Swiftvla: Unlocking spatiotemporal dynamics for lightweight vla models at minimal overhead. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13474–13485, June 2026.

Fei Ni, Jianye Hao, Shiguang Wu, Longxin Kou, Jiashun Liu, Yan Zheng, Bin Wang, and Yuzheng Zhuang. Generate subgoal images before act: Unlocking the chain-of-thought reasoning in diffusion model for robot manipulation with multimodal prompts. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13991–14000. IEEE, 2024.

Rui Qian, Haozhi Cao, Tianchen Deng, Shenghai Yuan, and Lihua Xie. Splatssc: Decoupled depth-guided gaussian splatting for semantic scene completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 8520–8528, 2026.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. icarl: Incremental classifier and representation learning. In *2017 IEEE conference on computer vision and pattern recognition (CVPR)*, pp. 5533–5542. IEEE, 2017.

Kaushik Roy, Akila Dissanayake, Brendan Tidd, and Peyman Moghadam. M2distill: Multi-modal distillation for lifelong imitation learning. In *2025 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1429–1435. IEEE, 2025.

Shilin Shan, Chuhao Zhou, Ruize Wang, Xinyan Chen, Xiangyu Chen, Xinyu Zhou, Boyu Ma, Iris Yuxuan Hu, Jingliang Li, Celeste Yuxuan Hu, et al. Learning physical interaction: A survey of tactile-and force-aware robot learning. *arXiv preprint arXiv:2608.07558*, 2026.

Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pp. 785–799. PMLR, 2023.

Wenxuan Song, Ziyang Zhou, Han Zhao, Jiayi Chen, Pengxiang Ding, Haodong Yan, Yuxin Huang, Feilong Tang, Donglin Wang, and Haoang Li. Reconvla: Reconstructive vision-language-action model as effective robot perceiver. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 18549–18557, 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

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

Zhou Xian, Nikolaos Gkanatsios, Theophile Gervet, Tsung-Wei Ke, and Katerina Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *7th Annual Conference on Robot Learning*, 2023.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. *arXiv preprint arXiv:1904.00962*, 2019.

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.

Michał Zawalski, William Chen, Karl Pertsch, Oier Mees, Chelsea Finn, and Sergey Levine. Robotic control via embodied chain-of-thought reasoning. *arXiv preprint arXiv:2407.08693*, 2024.

Yanjie Ze, Ge Yan, Yueh-Hua Wu, Annabella Macaluso, Yuying Ge, Jianglong Ye, Nicklas Hansen, Li Erran Li, and Xiaolong Wang. Gnfactor: Multi-task real robot learning with generalizable neural feature fields. In *Conference on Robot Learning*, pp. 284–301. PMLR, 2023.

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

Wei Zhao, Pengxiang Ding, Zhang Min, Zhefei Gong, Shuanghao Bai, Han Zhao, and Donglin Wang. Vlas: Vision-language-action model with speech instructions for customized robot manipulation. In *International Conference on Learning Representations*, volume 2025, pp. 51676–51693, 2025.

Zexin Zheng, Jia-Feng Cai, Xiao-Ming Wu, Yi-Lin Wei, Yu-Ming Tang, Ancong Wu, and Wei-Shi Zheng. imanip: Skill-incremental learning for robotic manipulation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 13890–13900. IEEE, 2025.

Yifeng Zhu, Peter Stone, and Yuke Zhu. Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation. *IEEE Robotics and Automation Letters*, 7(2):4126–4133, 2022.

APPENDIX CONTENTS

In our appendix, we elaborate on the details of the following aspects:

- A. Implementation Details
- B. Additional Experimental Results
- C. Hyperparameter Analysis
- D. Real-World Setup

A IMPLEMENTATION DETAILS

In this section, we detail the benchmark details and method implementation details. For benchmark details, we elaborate on the structure of each incremental setting, skill definition, and variations. In terms of method implementation details, we first introduce the universal settings that apply to all methods, and then provide the method-specific modifications. Code can be found in our supplementary materials.

A.1 BENCHMARK DETAILS

As mentioned in Section 4, we follow the benchmark protocols of iManip (Zheng et al., 2025) to re-implement the incremental benchmark. For visual observation, we adopt RGB-D images from a single front camera with a resolution of 128×128 as the universal input for all backbones. The RGB-D images are then projected as colorful 3D voxels, which are further encoded by a 3D convolutional network before input into the PercieverIO. Specifically for GNFactor and ManiGaussian, which contain 3D rendering modules that require multi-view images as supervision signals, the same setting of 20 cameras is used specifically for their rendering modules. Linguistic instructions are encoded by CLIP RN50 (Radford et al., 2021), while the proprioception data is encoded by an MLP layer. Following the backbone settings (Shridhar et al., 2023; Ze et al., 2023; Lu et al., 2024), we utilize the same discrete presentation for position and rotation formatting, where $\mathbf{a}_t^{\text{pos}} \in \mathbb{R}^{100^3}$ is a one-hot matrix that represents a 1.0m^3 volume indicating the voxel where the end-effector position would approach and $\mathbf{a}_t^{\text{rot}} \in \mathbb{R}^{(360/5)\times 3}$ denotes one-hot rotation GT with a discretization of 5°. During training and evaluation, the workspace in simulation is bounded from $[-0.3, -0.5, 0.6]$ to $[0.7, 0.5, 1.6]$ m (x, y, z).

Table 5: Task variations and train/test coverage.

ID	Skill	Variations	Variation semantics
1	close_jar	20	Target jar color
2	open_drawer	3	Bottom, middle, or top drawer
3	sweep_to_dustpan_of_size	2	Tall or short dustpan
4	meat_off_grill	2	Chicken or steak
5	turn_tap	2	Left or right tap
6	slide_block_to_color_target	4	Green, blue, pink, or yellow target
7	put_item_in_drawer	3	Bottom, middle, or top drawer
8	reach_and_drag	20	Target region color
9	push_buttons	50	Ordered colored-button sequence, requiring 1–3 presses
10	stack_blocks	60	20 target colors $\times$ stack heights of 2, 3, or 4

The incremental benchmark is composed of 10 tasks listed in Table 5. Under each incremental setting of B*n*-*k*N*m*, the skills of ID 1-*n* are utilized for base pre-training, while the following stages sequentially take *m* tasks per stage. For all runs, we generate the training demonstrations and define evaluation rollouts using the same random seed (seed=0) for fair comparison. Most incremental settings follow the same iteration protocols, with 100,000 iterations allocated for base pre-training

Table 6: Per-layer architecture of the chain predictor.

Layer / Component	**Input → Output**	**Function**
Perceiver latent pooling	$2048 \times 512 \rightarrow 1 \times 512$	Mean-pools the final Perceiver latent tokens into one scene token.
Language projection	$77 \times 128 \rightarrow 77 \times 512$	Projects the preprocessed CLIP token sequence to the chain-feature dimension.
Scene-language context	$(1 + 77) \times 512$	Concatenates the pooled scene token and projected language tokens.
Keyframe queries	7×512	Uses seven ordered learnable query slots; valid leading slots decode consecutive future actions, with the last valid slot assigned to the goal.
Cross-attention	$7 \times 512 \rightarrow 7 \times 512$	Uses four-head attention to extract information from the 78 scene-language tokens.
Cross-attention LayerNorm	$7 \times 512 \rightarrow 7 \times 512$	Applies residual addition followed by feature normalization.
Causal self-attention	$7 \times 512 \rightarrow 7 \times 512$	Models temporal dependencies while restricting each query to itself and earlier slots.
Self-attention LayerNorm	$7 \times 512 \rightarrow 7 \times 512$	Applies residual addition followed by feature normalization.
Translation head	$512 \rightarrow 3 \times 100$	Predicts categorical voxel distributions independently for the x, y, and z axes.
Chain output	$7 \times 3 \times 100$	Represents the translation logits for all predicted future keyframes.
Chain-context pooling	$7 \times 512 \rightarrow 1 \times 512$	Pools detached query features using normalized weights proportional to 0.7^i.
Decoder injection	$2048 \rightarrow 2049$ context tokens	Appends the pooled chain token to the original decoder cross-attention context.

and 50,000 used for incremental learning per stage. The only exception is B2-4N2, where we reduce the base iterations to 50,000 since it involves much fewer base tasks during the pre-training stage. While iManip (Zheng et al., 2025) mentions that it utilizes 100,000 iterations for training, it does not explicitly point out whether this is for base pre-training or incremental update. Our 50,000-step setting is therefore deduced from its ablation study in Table 3 and our empirical estimation based on the number of tasks for each stage.

RLBench skills with varying task configurations. `open_drawer` and `put_item_in_drawer` each contain three variations corresponding to the bottom, middle, and top drawers; `sweep_to_dustpan_of_size`, `meat_off_grill`, and `turn_tap` each contain two variations, respectively specifying the dustpan size, meat type, and tap side; and `slide_block_to_color_target` contains four color-target variations. The more diverse tasks include `close_jar` and `reach_and_drag`, each with 20 target-color variations; `push_buttons`, with 50 variations combining ordered colored-button sequences and one to three required presses; and `stack_blocks`, with 60 variations combining 20 target colors with stack heights of two, three, or four blocks. In total, these tasks span 168 predefined skill variations. Each episode samples from the corresponding task's full variation space, although the finite training and evaluation sets do not necessarily contain every variation for high-cardinality tasks.

A.2 Method Implementation Details

In this section, we first detail the specific design of GROW and our hyperparameter selection. Subsequently, we introduce the implementation of all baselines.

A.2.1 Implementation Details of GROW

Hyperparameter setting. GROW introduces three major components to each keyframe-based backbone during both base pre-training and incremental stages, where most of the specifications remain identical. For dual-horizon grounding, both Gaussian anchors $\mathbf{g}_t^{\text{next}}$ and $\mathbf{g}_t^{\text{goal}}$ share the same configuration with $\sigma = 3$ (Eq. 5). The corresponding coefficient λ_{n} and λ_{g} are both set as 0.01 (Eq. 6). As for the action chain predictor, the length of learnable queries $\mathcal{Q}_{\text{chain}}$ is set to $k = 7$, while the decay factor τ is set to 0.7 (Eq. 8, Eq. 10, Eq. 11, and Eq. 12). In terms of chain-based replay herding, we allocate a budget of $B_m = 2$ for each previous skill per average keyframe (Eq. 14). The loss coefficient λ_{dis} is 0.01 while λ_{chain} is set as 0.001. We conduct a series of sensitivity analyses across some important hyperparameters.

Architecture of Action Chain Predictor. We list the per-layer components of the action chain predictor in Table 6 along with their corresponding function. Overall speaking, the proposed action chain predictor is an intuitive PerceiverIO-similar module, using learnable queries to decode the action chain features. It is also a lightweight component, containing about 2.3M trainable parameters, which additionally occupy 35.5MB GPU memory ($+0.1\%$ compared to the TIB baseline) during the training process. Despite being simple and intuitive, enabling the action chain predictor solely can lead to substantial improvement when using our GROW or TIB baseline as we will show in the following section.

Details of LoRA-based Update. Here we elaborate on the LoRA-based update we used for GROW. At each incremental stage, we initialize a fresh rank-8 LoRA adapter ($\alpha = 16$) on all PerceiverIO encoder cross-attention ($\texttt{CrossAttn}_{\text{enc}}$ in Eq. 1), latent self-attention ($\texttt{SelfAttn}_{\times M}$ in Eq. 2), and decoder cross-attention projections ($\texttt{CrossAttn}_{\text{dec}}$ in Eq. 2). The voxel encoder and original PerceiverIO parameters are frozen, including their batch-normalization statistics, while the LoRA parameters, action decoder, chain predictor, and optional neural renderer from ManiGaussian and GNFactor are trained for 50,000 iterations. The learned LoRA updates are then merged into the corresponding attention weights via summation, after which only the merged attention projections are re-calibrated for 2,000 iterations using a $0.1\times$ learning rate. During the re-calibration stage, the decoder and chain predictor remain frozen, and rendering and distillation are disabled. The merged and recalibrated model becomes the initialization for the next stage, where a new zero-delta LoRA adapter is introduced. In this way, we can maintain a finite size of trainable parameters while incrementally learning new skills.

A.2.2 Implementation Details of Baselines

We first introduce the universal setting shared across all baselines to make a fair comparison with GROW, and then detail the baseline-specific settings individually.

Universal Setting. For all baselines, we implement them based on their original paper and adopt the same incremental training setting. Specifically, following the training protocols used for PerceverIO-based backbones (Ze et al., 2023; Lu et al., 2024), all runs, including GROW, are trained using the LAMB optimizer (You et al., 2019) with an initial learning rate of 5×10^{-4}. Experiments are conducted with a batch size of 1, using a single NVIDIA RTX PRO 6000 GPU. Below are the baseline-specific configurations. For each backbone, all incremental training starts with the same base checkpoint (except for GROW, as it introduces addition modules during the base training) to make a fair comparison. We use the same fixed seed for all methods to ensure identical training demonstrations and evaluation episodes. The reported results therefore characterize this controlled benchmark instance rather than training-seed variance.

- **Seq-FT.** This is the intuitive sequential fine-tuning using each backbone. For each incremental stage, the backbone is fine-tuned using the optimizer and checkpoint inherited from the end of the previous stage. Only the training data from the unseen skills are observable, and no replays are involved for each incremental stage.
- **EWC.** We implement EWC using the canonical Online-EWC formulation (Kirkpatrick et al., 2017) without replaying old-task demonstrations. Each incremental stage is trained only on the newly introduced task while all BC-policy parameters remain trainable. After each stage, an empirical diagonal Fisher matrix is estimated from the squared gradients of the action-only behavioral-cloning loss over random 1,000 valid transitions from that stage.

The Fisher statistics are accumulated as $\bar{F}_k = \gamma \bar{F}_{k-1} + F_k$, with $\gamma = 0.9$, and the final policy parameters are stored as the new anchor. At the next stage, deviations from this anchor are penalized using a Fisher-weighted quadratic regularizer with $\lambda_{\mathrm{EWC}} = 100$. The protected scope covers the active voxel encoder, language and proprioceptive processing, Perceiver transformer, and action decoder, while excluding the frozen language model and neural renderer. Batch-normalization statistics are frozen during incremental training.

- **TAIL.** Our task-agnostic TAIL implementation uses the jointly trained base policy as an immutable shared backbone. For every incremental task, an independent expert is initialized directly from this base policy, rather than from the preceding incremental expert. Each expert trains a fresh rank-8 LoRA adapter with $\alpha = 16$ on all PerceiverIO attention projections together with a complete task-specific action decoder, while the remaining backbone parameters are frozen. TAIL uses only behavioral cloning on the current task. At evaluation, no task identity is used to select an expert. The primary task-agnostic result uniformly averages the success rates obtained by the stage-0 policy and all available task experts, while an additional probability-ensemble variant averages their normalized translation, rotation, gripper, and collision distributions at every action step.
- **TIB.** TIB is a keyframe-based replay baseline introduced in iManip (Zheng et al., 2025), which retains a compact set of representative old-task keyframe transitions and combines exemplar replay with output distillation. After each stage, every transition is represented by a 220-dimensional action feature formed by concatenating the predicted rotation/gripper probabilities and collision probabilities. Transitions are grouped according to their ordinal keyframe position, L2-normalized, and selected independently within each group using greedy mean herding. We retain 2 transitions per keyframe position by default to form a fair comparison with GROW using the same amount of replay budget (i.e., $B_m = 2$). During the following stage, all new-task transitions are trained jointly with the selected old-task exemplars using task-uniform replay. A frozen copy of the preceding policy provides an old-task distillation target, with an MSE loss between the old and current translation, rotation, gripper, and collision probabilities weighted by $\lambda_{\mathrm{dis}} = 0.01$.
- **iManip.** iManip (Zheng et al., 2025) expands the Perceiver architecture at every incremental stage while freezing previously learned components. Following their protocol in the manuscript, for each stage, we introduce 16 new learnable prompt tokens on the input/context side of the encoder cross-attention and append a 512×8 query extension and a 512×8 key extension to every latent self-attention layer. With eight attention heads, this increases the per-head query/key dimension by one at each stage, while leaving the value projections, output projections, and 2048×512 latent array unchanged. Previously learned prompts and query/key extensions are frozen, and only the newly added parameters and action decoder are optimized. Old-task memory is constructed using farthest-distance entropy sampling: transitions are grouped by keyframe position, scored by their total action cross-entropy over translation, rotation, gripper, and collision predictions, and selected to cover diverse uncertainty levels. Incremental training combines these exemplars with the complete new-task data and applies old-task action-probability distillation with weight $\lambda_{\mathrm{dis}} = 0.01$.

Reproduction Details and Performance Gap. We observe that the reproduced iManip performance is lower than that reported in the original paper despite following its description. Several implementation details make an exact reproduction difficult. First, the published iManip model is described as a standalone Extendable PerceiverIO architecture, while an exact layer-wise mapping between this model and the PerAct-/ManiGaussian-style backbones used in its comparisons is not fully specified. We therefore implement the reported incremental mechanisms consistently on top of each backbone in our evaluation rather than reproducing an architecture specific to iManip.

Second, the original paper reports a batch size of 1, learning rate of 2×10^{-3}, and 100k training iterations (Zheng et al., 2025). Its freezing analysis further explicitly trains each B5-1N1 experiment for 100k iterations and reports convergence at approximately 60k iterations for the selected configuration. However, the exact per-stage optimization schedule for the full B5-5N1 sequence is not separately documented. For a controlled comparison, we therefore adopt a fast tryout run on stage 1 of B5-5N1, using the same stage-wise training schedule for all incremental methods in our experiments. Importantly, directly applying the reported learning rate 2×10^{-3} to our iManip

Table 7: Performance of iManip using different learning rates.

Learning Rate	Base	Stage 1		
		Old	New	All
0.002	60.0	8.0	16.0	9.3
0.0005	60.0	33.6	28.0	32.7

Table 8: Baseline TIB with GROW's components. Here * indicates disabling the normalization statistics freezing. "Chain" represents the chain predictor. All runs tune only the action decoder during the incremental stages.

Method	Steps	Base	Stage 1		
			Old	New	All
MG-A	50k+10k	46.4	6.4	16.0	8.0
MG-N	50k+10k	46.4	12.8	4.0	11.3
MG-TIB*	50k+10k	46.4	3.2	12.0	4.7
MG-TIB	50k+10k	46.4	24.8	0.0	20.7
MG-TIB	50k+50k	46.4	13.6	8.0	11.3
MG-TIB+$\mathcal{L}_{\mathrm{dh}}$	50k+10k	44.8	34.4	24.4	32.7
MG-TIB+Chain	50k+10k	44.8	30.4	8.0	26.7
MG-TIB+$\mathcal{L}_{\mathrm{dh}}$+Chain	50k+10k	45.6	34.4	**28.0**	**33.3**
MG-TIB+$\mathcal{L}_{\mathrm{dh}}$	50k+50k	44.8	16.0	20.0	16.7
MG-TIB+Chain	50k+50k	36.8	24.0	**28.0**	24.7
MG-TIB+$\mathcal{L}_{\mathrm{dh}}$+Chain	50k+50k	42.4	**35.2**	16.0	32.0

implementation results in substantially lower performance (e.g., 9.3% overall success at Stage 1), whereas reducing it to 5×10^{-4} (the default learning rate shared across all PerceiverIO-based backbones) improves the corresponding performance to 32.7% (Table 7). We consequently report the latter, stronger configuration for iManip throughout our experiments.

Finally, we retain the original farthest-distance entropy replay strategy and its budget of two samples per keyframe position. With such a limited budget, this diversity-oriented criterion can be sensitive to the distribution of action-prediction uncertainty, since only a small number of widely separated samples are retained. Nevertheless, we preserve the original selection rule rather than replacing it with an alternative herding strategy. Therefore, our iManip results should be interpreted as a controlled reimplementation of its reported incremental mechanisms under our common backbone and training protocol.

B Additional Experimental Results

In this section, we first introduce some additional quantitative results to better justify the effectiveness of GROW, and then include more qualitative comparisons.

B.1 Quantitative Results

B.1.1 GROW's Components with TIB Baseline

We extend the key components with vanilla TIB baselines under different base and incremental iterations to form additional ablation studies as in Table 8. Specifically, all runs tune only the action decoder during the incremental stages, which means our addition $\mathcal{L}_{\mathrm{dh}}$ and action chain predictor are only functional during the base pre-training stage. Even so, adding either $\mathcal{L}_{\mathrm{dh}}$ or the action chain predictor can lead to a non-trivial improvement of more than 6.0% and 5.4% when incrementally updating with 10k and 50k iterations, respectively. This indicates the robustness of the key components in GROW under various incremental settings.

B.1.2 GROW with Different Update Mechanisms

We further combine GROW with various updating mechanisms, including parameter freezing, universal LoRA, and EWC using Fisher information regularization. As shown in Table 9, using each

Table 9: GROW and TIB with different incremental update protocols. Here "F" indicates freezing encoder and PercieverIO's components and only update the action decoder, while "L" denotes the universal LoRA during incremental stages.

Method	Base	Stage 1			Stage 2			Stage 3			Stage 4			Stage 5			Avg		
		Old	New	All	Old	New	All	Old	New	All	Old	New	All	Old	New	All	Old	New	All
TIB-F	**60.0**	28.0	8.0	24.7	21.3	**16.0**	20.6	18.9	16.0	18.5	25.5	**36.0**	26.7	16.0	0.0	14.4	21.9	15.2	21.0
TIB-L	**60.0**	14.4	20.0	15.3	29.3	**16.0**	27.4	21.7	28.0	22.5	16.0	8.0	15.1	15.1	0.0	13.6	19.3	14.4	18.8
GROW-F	59.2	**40.8**	**36.0**	**40.0**	36.7	0.0	31.4	30.9	40.0	32.0	22.0	20.0	21.8	20.0	0.0	18.0	30.1	19.2	28.6
GROW-L	59.2	40.0	32.0	38.7	29.3	**16.0**	27.4	**35.4**	40.0	**36.0**	25.5	28.0	25.8	22.2	0.0	20.0	30.5	**23.2**	29.6
GROW-EWC	59.2	38.4	**36.0**	38.0	25.3	4.0	22.3	21.1	40.0	23.5	26.0	12.0	24.4	28.9	0.0	26.0	28.0	18.4	26.8
TIB	**60.0**	16.0	28.0	18.0	28.7	0.0	24.6	19.4	**56.0**	24.0	18.0	8.0	16.9	25.3	0.0	22.8	21.5	18.4	21.3
iManip	**60.0**	33.6	28.0	32.7	24.0	**16.0**	22.9	16.0	24.0	17.0	22.0	12.0	20.9	16.0	0.0	14.4	22.3	16.0	21.6
GROW	59.2	40.0	**36.0**	39.3	**41.3**	**16.0**	**37.7**	25.1	28.0	25.5	**33.5**	16.0	**31.6**	**30.2**	0.0	**27.2**	**34.0**	19.2	**32.3**

Table 10: Stage-wise performance ablation studies in Table 2 with corresponding ablation IDs.

	ID	Base	Stage 1			Stage 2			Stage 3			Stage 4			Stage 5			Avg		
			Old	New	All	Old	New	All	Old	New	All	Old	New	All	Old	New	All	Old	New	All
Base + Incremental	1	47.2	34.4	24.0	32.7	12.0	32.0	14.9	6.9	80.0	16.0	17.5	8.0	16.4	24.0	4.0	22.0	19.0	29.6	20.4
	2	52.0	26.4	28.0	26.7	28.0	8.0	25.1	21.7	24.0	22.0	14.0	8.0	13.3	20.4	0.0	18.4	22.1	13.6	21.1
	3	53.0	32.0	16.0	29.3	25.3	12.0	23.4	22.9	20.0	22.5	22.0	8.0	20.4	20.0	0.0	18.0	24.4	11.2	22.7
	5	54.4	31.2	36.0	32.0	28.7	12.0	26.3	26.3	40.0	28.0	17.5	4.0	16.0	18.7	0.0	16.8	24.5	18.4	23.8
	6	55.3	29.6	24.0	28.7	30.7	8.0	27.4	25.1	36.0	26.5	20.5	8.0	19.1	19.1	0.0	17.2	25.0	15.2	23.8
	7	48.8	36.0	12.0	32.0	32.7	4.0	28.6	34.3	24.0	33.0	23.5	8.0	21.8	16.9	0.0	15.2	28.7	9.6	26.1
Incremental	1	59.2	39.2	28.0	37.3	33.3	4.0	29.1	21.1	28.0	22.0	25.0	8.0	23.1	20.9	0.0	18.8	27.9	13.6	26.1
	2	59.2	36.8	52.0	39.3	38.7	0.0	33.1	33.1	36.0	33.5	30.5	16.0	28.9	25.3	0.0	22.8	32.9	20.8	31.5
	3	59.2	36.8	40.0	37.3	33.3	16.0	30.9	26.3	36.0	27.5	17.5	24.0	18.2	20.9	4.0	19.2	27.0	24.0	26.6
	4	59.2	39.2	20.0	36.0	33.3	12.0	30.3	34.3	20.0	32.5	19.5	16.0	19.1	20.4	0.0	18.4	29.3	13.6	27.3
	5	59.2	40.8	40.0	40.7	31.3	8.0	28.0	33.7	12.0	31.0	26.5	16.0	25.3	24.4	0.0	22.0	31.3	15.2	29.4
	6	59.2	40.0	16.0	36.0	38.0	4.0	33.1	3.4	44.0	8.5	30.0	20.0	28.9	22.7	0.0	20.4	26.8	16.8	25.4
	7	59.2	23.2	28.0	24.0	30.0	0.0	25.7	25.1	24.0	25.0	23.5	24.0	23.6	19.6	0.0	17.6	24.3	15.2	23.2

updating mechanism with GROW can lead to SOTA performance compared with previous TIB and iManip. One interesting observation is that the universal LoRA version of GROW (GROW-L) achieves the best average new task comprehension across all variants, surpassing vanilla GROW by 4.0% on average. Nevertheless, our stage-wise LoRA with re-calibration strikes a better balance between old skill preservation and new skill comprehension, and therefore we adopt this strategy for all other experiments. Additionally, naively combining LoRA or freezing the PerceiverIO cannot achieve noticeable improvements on TIB, which demonstrates the improvement brought by the key components of GROW.

B.1.3 Stage-wise Results of Ablation Studies

We further supplement the stage-wise results in Table 10 that lead to the overall ablation studies in Table 2.

B.2 Qualitative Visualization

B.2.1 More Comparison of Attention Heatmaps

Figure 5 supplements more heatmap visualizations comparing GROW with iManip. We specifically visualize the heatmap of the final-stage checkpoints against their attention focus with regard to the base tasks (task IDs 2-5 as in Table 5). For all shown tasks and threshold levels, GROW exhibits more compact and semantically meaningful activation patterns, remaining concentrated around the objects and regions directly involved in the skill execution. By contrast, iManip often allocates a noticeable portion of its high-response attention to the upper corner of the voxel space, away from the actual manipulation region. This phenomenon indicates that, after repeated incremental updates, GROW better preserves task-relevant perception grounding, while iManip suffers more from attention drift.

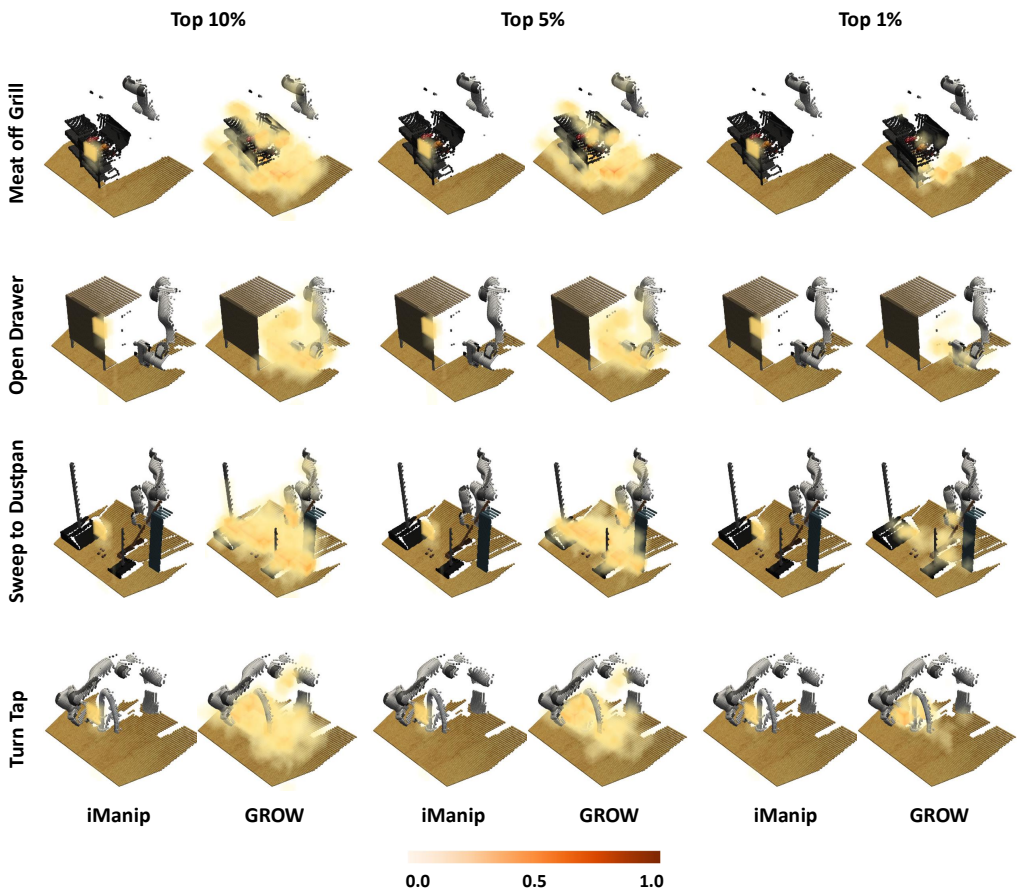

Figure 5: More comparison of attention heatmaps between iManip and GROW. All results are from the checkpoints of Stage 5 .

B.3 Simulation Rollout Videos

To further qualitatively evaluate long-term skill retention, we provide 10 simulation rollout videos collected using the final Stage 5 checkpoints of iManip and GROW in our supplementary materials. All videos are evaluated on the base skills that were learned before incremental adaptation, allowing direct comparison of how well previously acquired behaviors are preserved after the complete five-stage learning sequence. The paired rollouts highlight differences in execution stability, task-relevant interaction, and failure modes between the two methods. In particular, GROW more consistently maintains coherent skill execution on these previously learned tasks, while iManip exhibits more frequent degradation after long-sequence incremental updates. These qualitative results complement the quantitative evaluation by illustrating how differences in skill-level retention manifest during actual policy rollout.

C Hyper-Parameter Analysis

Table 11 studies the sensitivity of GROW to the loss coefficients of next-step grounding λ_{next}, goal grounding λ_{goal}, and action-chain prediction λ_{chain}, together with the Gaussian variance σ used for spatial grounding. Note that we incrementally train GROW with Stage 1 using the same B5-5N1 base checkpoint to form a fast validation. Overall, GROW remains reasonably robust across a broad range of settings, although excessively large grounding or chain weights can noticeably degrade the balance between old-skill retention and new-skill acquisition. Notably, $\lambda_{\text{next}} = 0.03$ and $\lambda_{\text{goal}} = 0.05$ achieve higher performance than our default configuration in this stage-wise analysis, reaching 44.0% overall success compared with 39.3%. We further evaluated both configurations over the complete incremental-learning sequence; however, their improvements did not

Table 11: Hyper-parameter analysis of GROW.

Hyper-parameter	Setting	Old	New	All
λ_{next}	0.03	**45.6**	**36.0**	**44.0**
	0.05	40.0	20.0	36.7
	0.07	33.6	32.0	33.3
	0.10	32.8	28.0	32.0
λ_{goal}	0.03	30.4	32.0	30.7
	0.05	**45.6**	**36.0**	**44.0**
	0.07	36.0	36.0	36.0
	0.10	37.6	4.0	32.0
λ_{chain}	0.003	35.2	32.0	34.7
	0.005	**37.6**	28.0	**36.0**
	0.007	34.4	**36.0**	34.7
	0.010	33.6	24.0	32.0
σ	1	35.2	12.0	31.3
	2	28.8	28.0	28.7
	4	**34.4**	**36.0**	**34.7**
	5	33.6	24.0	32.0
GROW	-	40.0	36.0	39.3

persist across later stages, and both resulted in inferior overall performance compared with our default setting. We therefore retain the default hyperparameters used throughout the main experiments, prioritizing stable performance across the full incremental sequence rather than the best result at an individual stage. This analysis is conducted using the ManiGaussian backbone, while the same default hyperparameters are directly adopted for PerAct and GNFactor without backbone-specific tuning. Their reported performance may therefore be further improved with dedicated hyperparameter optimization.

D Real-World Setup

In this section, we introduce the setup of our real-world experiments, including hardware setup, task definition and postprocessing, evaluation rollouts, and the qualitative results.

D.1 Hardware setup

The real-world platform consists of a seven-degree-of-freedom Franka FR3 manipulator equipped with a parallel-jaw gripper. Manipulation takes place in a tabletop workspace containing the objects required for block stacking, placing items into a basket, and sliding a block onto a colored target. The robot base provides the common reference frame for describing the workspace, camera geometry, and end-effector states.

Visual observations are obtained from a fixed, external Intel RealSense D455 RGB-D camera overlooking the manipulation area. The experiments use a single observation viewpoint only. The D455 camera is configured to provide RGB images at 640×480 pixels and depth images at 848×480 pixels, both at a nominal frequency of 15 Hz. Camera synchronization, depth-to-color alignment, and colored point-cloud generation are enabled. We perform eye-on-base calibration between the external camera and the robot using the `easy_handeye` package[1].

Data collection is done by teleoperation with Meta Quest 2, with most of the code adopted from `franka_vr`[2]. To facilitate more effective and convenient teleoperation, the controller buttons

[1] https://github.com/IFL-CAMP/easy_handeye

[2] https://github.com/ZorAttC/franka_vr

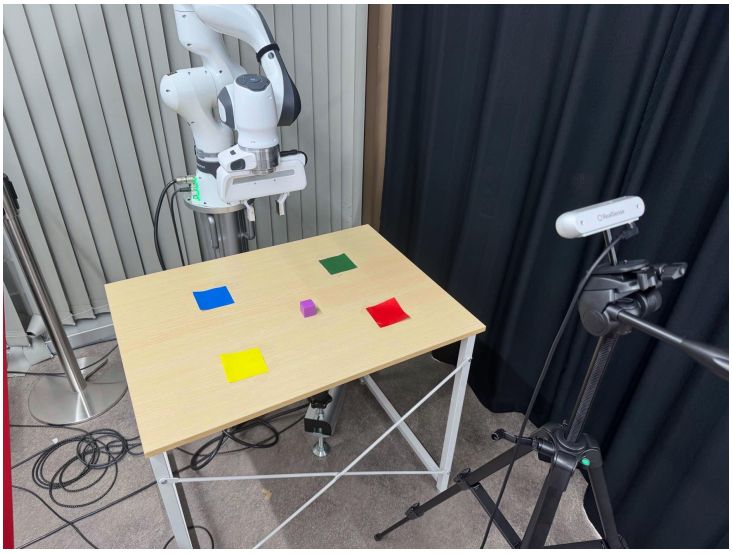

Figure 6: Hardware setup for real-world experiments.

provide translation-only, rotation-only, yaw-only, and roll/pitch-only modes. Switching these constraints re-establishes the reference at the current robot and controller poses to avoid returning to an older reference. A clutch command similarly re-establishes the reference without commanding arm movement. A precision mode reduces both translational and angular velocity limits to 35% of their normal values. Together, we can operate the FR3 to do some precise movement, manipulating small objects such as blocks and a toy pear. We attach a teleoperation video in the supplementary materials for reference. Robot and camera streams are recorded separately as ROS bags. Robot recordings include arm joint states, gripper joint states, and dynamic and static transforms; camera recordings contain the enabled sensor streams. Recorded timestamps support subsequent temporal alignment and reconstruction of end-effector trajectories. This preserves measured robot states independently of the operator's requested targets.

D.2 Task Definition and Postprocessing

Task Definition and Variations. Our real-world experiments involve three tasks, including `put_items_in_baskets`, `stack_blocks`, and `slide_blocks_to_regions`. Each collected demonstration has a corresponding record in our `*.csv` table in the supplementary materials, including the linguistic instruction, environmental descriptions, and ROS bag ID. Specifically, for `stack_blocks` the 20 demonstrations cover different ordered color pairs among red, blue, and green blocks, requiring the policy to distinguish both the source and target block while operating in a consistent two-block scene. For `put_items_in_baskets`, the demonstrations vary both object identity (red/green apple, strawberry, and pear) and target basket color, while progressively increasing scene complexity from one object with two baskets, to two objects with two baskets, and finally four objects with three baskets. For `slide_blocks_to_regions`, the demonstrations vary the instructed target color as well as the set of candidate colored regions in the scene, including combinations of red, green, blue, yellow, orange, and black targets.

Postprocessing and Training. The postprocessing procedures extract the raw ROS bags into keyframes used to train PerAct models for each incremental method. All geometric observations and action targets are first expressed in the robot base frame. Camera-frame points are transformed into the base frame using the calibrated camera-to-base transformation. Non-finite points and points outside the configured workspace are removed. The workspace bounds are $x \in [0.1, 1.2]$, $y \in [-0.5, 0.6]$, and $z \in [-0.12, 0.88]$ metres. Each retained observation is represented by 16,384 paired XYZ–RGB samples. Deterministic random sampling is performed without replacement when sufficient valid points are available and with replacement otherwise. The sampled coordinates and

colors are stored as $128 \times 128 \times 3$ arrays for compatibility with the downstream data interface. This arrangement is a storage convention and does not preserve the original camera-image pixel layout.

For keyframe extraction, considering the fact that teleoperation usually involves small but repetitive local motion when targeting small objects, we adopt greedy pruning to remove consecutive and similar keyframes. Specifically, keyframes are detected from robot stops, gripper-state changes, and the terminal observation. Stop detection uses a joint-velocity tolerance of 0.02. Redundant stopped poses are subsequently pruned using translation and rotation thresholds of 1 cm and 5°, respectively. Gripper-change and terminal anchors are preserved during this pose-based pruning.

As for the training details, we adopt the identical hyperparameter settings and configurations as in the simulations, forming a fair re-implementation from simulation to real-world rollouts.

D.3 Evaluation Rollouts

The evaluation rollouts are conducted following some universal protocols. We try our best to maintain the same environmental conditions for evaluation across all methods (e.g., object positions and lighting conditions), although there might be some inevitable minor differences across each rollout. The environment conditions are uniformly sampled from the training data with different object positions. The online prediction is generated using a single RTX 4090 GPU under a keyframe prediction paradigm: the agent captures the observation when the end-effector point reaches the previous target pose, and then predicts the next end-effector pose for execution. A mannual buffer time of 3s is set between executing the predicted end-effector pose so that user can check the validity of the predicted pose before execution. The rollouts that exceed 25 steps without completing the skill are regarded as a failure.

D.4 Qualitative Results

We include three pairs of rollout videos in the supplementary materials, each comparing iManip and GROW on the same real-world manipulation skill using our robotic platform. The videos qualitatively illustrate how the differences in skill preservation and task understanding manifest during execution. For `stack_blocks`, GROW accurately identifies the instructed target and successfully stacks the green block on the blue block. In contrast, iManip approaches an incorrect placement location slightly offset from the blue block, which leads to a failed stacking attempt. For `put_items_in_baskets`, iManip fails to localize and grasp the instructed apple reliably, whereas GROW successfully grasps the target object and places it into the corresponding basket. For `slide_blocks_to_regions`, iManip remains largely inactive after initialization and fails to execute the instructed behavior, while GROW correctly interprets the task and successfully slides the block toward the designated color target. These rollout examples complement the quantitative results by showing that GROW preserves more reliable instruction-conditioned perception and execution in real-world settings, particularly when accurate object localization and task-level action progression are required.