CLARE: Continual Learning for Vision-Language-Action Models via Autonomous Adapter Routing and Expansion

Ralf Römer*,1 Yi Zhang*,1 Yuming Li[1] Angela P. Schoellig[1,2,3]

***Abstract*— To teach robots complex manipulation tasks, a common approach is to fine-tune a pre-trained vision-language-action model (VLA) on task-specific data. However, since this recipe updates existing representations, it is unsuitable for long-term operation in the real world, where robots must continually adapt to new tasks and environments while retaining the knowledge they have already acquired. Existing continual learning methods for robotics commonly require storing previous data (exemplars), struggle with long task sequences, or rely on task identifiers for deployment. To address these limitations, we propose CLARE, a general, parameter-efficient framework for exemplar-free continual learning with VLAs. CLARE introduces lightweight modular adapters into selected VLA modules and autonomously expands the model only where necessary when learning a new task, guided by layer-wise feature similarity. During deployment, an autoencoder-based routing mechanism dynamically activates the most relevant adapters without requiring task labels. Through extensive experiments on the LIBERO benchmark and five real-world tasks, we show that CLARE achieves high performance on new tasks without catastrophic forgetting of earlier tasks, significantly outperforming even exemplar-based methods. Code, data, and videos are available at our website: tum-lsy.github.io/clare.**

I. Introduction

Robots deployed in homes, hospitals, or warehouses must operate for long periods while facing ever-changing conditions and task demands. For instance, a household robot may need to operate a newly purchased appliance, or an assistive robot may meet patients with unfamiliar mobility profiles. In such settings, robots must continually acquire new skills without sacrificing previously acquired capabilities. This long-term adaptability, known as continual or lifelong learning [1], remains an open challenge in robotics despite decades of research [2]–[4].

Recent advances in vision-language-action models (VLAs) [5]–[8] have demonstrated strong performance on complex, long-horizon manipulation tasks by integrating perception, language understanding, and action generation within a unified model. Pre-training on internet-scale data and robot demonstrations [9], [10] provides VLAs with broad priors that enable some degree of generalization [6]. However, state-of-the-art VLAs still cannot adapt reliably to unseen tasks without fine-tuning on task-specific data [6]–[8]. In a continual learning setting, where new tasks and environments emerge over time, a naive approach would be to iteratively fully fine-tune (FFT) a VLA on new data. However, updating the parameters shared across modalities without regard for previously learned representations leads to significant degradation of both semantic grounding and policy performance on old tasks, a common issue known as catastrophic forgetting [11]. Therefore, current VLAs are not inherently capable of continual learning, hindering their deployment in non-stationary, real-world environments.

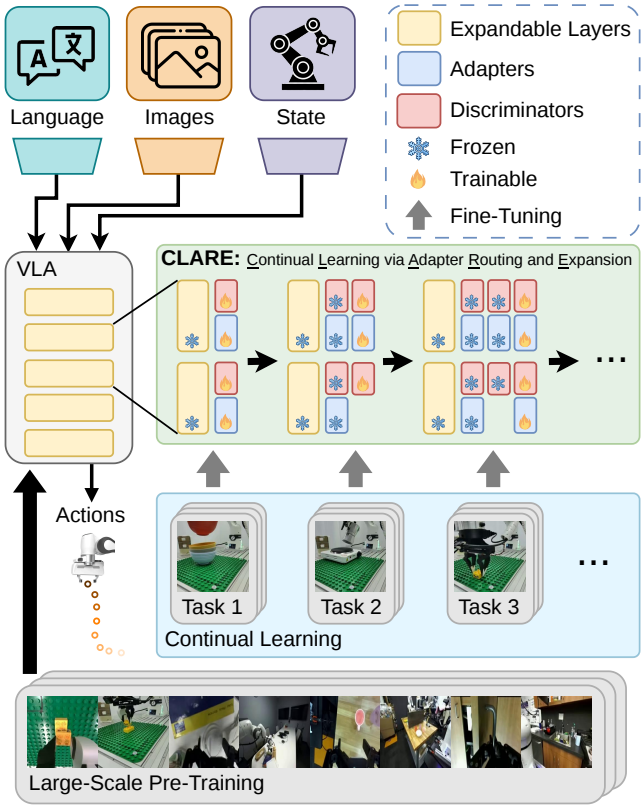


Fig. 1: CLARE autonomously and continually injects lightweight adapters into selected layers of a pre-trained vision-language-action model (VLA). During inference, the most relevant adapters are activated based on feature similarity, captured by learned discriminators. By fine-tuning only the newly added parameters at each stage, the policy acquires new task-specific knowledge without catastrophic forgetting of previously learned skills.

Experience replay (ER) [12]–[14] can mitigate forgetting but has significant practical drawbacks for robotics. Past data may be unavailable due to storage or privacy constraints in a lifelong learning setting. Maintaining and accessing a large replay buffer increases computational and memory overhead, and selecting representative samples for replay is

* Equal contribution.
[1] Technical University of Munich, Germany; TUM School of Computation, Information and Technology, Department of Computer Engineering, Learning Systems and Robotics Lab; Munich Institute of Robotics and Machine Intelligence (MIRMI).
[2] Robotics Institute Germany.
[3] Munich Center for Machine Learning.
Corresponding email: {ralf.roemer@tum.de}

challenging. For these reasons, there is a strong need for exemplar-free continual learning methods that enable VLAs to acquire new skills while preserving prior knowledge.

Modular and expandable architectures [15]–[17], which allocate new capacity for each task rather than overwriting shared representations, represent a promising direction for scalable continual learning. However, these methods often require oracle task identifiers, which are typically unavailable when robots operate autonomously in open-world settings. Moreover, the application of these methods to robotics has so far been mostly limited to multitask learning, where all tasks and data are available in advance [15], [18].

To close this gap, we introduce Continual Learning via Adapter Routing and Expansion (CLARE), a general framework that enables VLAs to continually incorporate new task-specific knowledge without exemplars, task labels, or predefined expansion rules. CLARE injects lightweight adapters into selected modules of the pre-trained model and expands only when the feature statistics indicate substantial novelty, as visualized in Figure 1. At deployment, an autoencoder-based routing mechanism dynamically selects among the adapters, enabling autonomous task-agnostic inference. This design maintains a balance between stability and plasticity by preserving pre-trained representations and adding capacity as needed, enabling VLAs to learn new tasks with minimal parameter growth. In summary, our main contributions are:

- A lightweight, modular framework enabling VLAs to learn new tasks without catastrophic forgetting.
- An autonomous routing mechanism that activates the most suitable adapters during inference using feature similarity without task identifiers.
- A dynamic expansion strategy that increases parameter count by only about 2% per task in our experiments.
- Extensive experiments in simulation and real-world settings demonstrating that CLARE significantly outperforms continual learning baselines.

While our evaluation focuses on a compact VLA architecture, the modular design of CLARE is architecture-agnostic and directly applicable to larger VLAs.

II. Related Work

1) Vision-Language-Action Models: In recent years, the robotics community has begun collecting massive multimodal datasets [10] and exploiting generative modeling architectures [19], [20] to equip robots with broad semantic priors and facilitate generalization across tasks and environments [5]–[8]. These VLAs are usually based on vision-language models [21] and pre-trained on robot demonstrations [10] via imitation learning, enabling end-to-end mapping from high-dimensional multimodal observations to robot actions. Despite the growing scale of training data and model capacity, the ability of VLAs to generalize zero-shot to unseen tasks and environments remains limited [6], [22]. These models often overfit to their pre-training domains, as real-world robot data is much more expensive and scarce than web-scale vision and text datasets [3]. Thus, it has become a common paradigm to fine-tune a pre-trained VLA on curated, high-quality demonstrations [5]–[8] to achieve high performance on a specific task. However, in settings where tasks arrive sequentially and old data may be unavailable, this naive fine-tuning recipe is inadequate, as it overwrites previously learned task knowledge, leading to catastrophic forgetting [23].

2) Continual Learning: Acquiring new skills from a stream of data without catastrophic forgetting of previously learned capabilities or losing plasticity is a hard problem in deep learning [4]. ER-based approaches [12]–[14] retain a subset of past examples and mix them with new data to preserve existing representations during training. Since storing exemplars may be infeasible, regularization-based methods [24], [25] constrain parameter updates for weights deemed important to a past task. Related to this idea, PackNet [26] prunes less relevant parameters from previously learned tasks and re-trains them for the new incoming data. However, methods like EWC or PackNet struggle with long task sequences as they are restricted by a fixed set of initial parameters. To overcome capacity bottlenecks and avoid catastrophic forgetting, architectural methods [15], [16], [27] inject new parameters or modules for novel tasks. By keeping the original model frozen and leveraging techniques such as low-rank adaptation (LoRA) [28], these methods can store new task-specific knowledge in a memory-efficient way.

3) Continual Imitation Learning in Robotics: Data scarcity and safety concerns make continual learning for robotics particularly challenging [2], [3]. LOTUS [29], a hierarchical method, constructs an ever-growing library of skill policies [30] and uses a meta-policy trained using ER for skill selection during deployment. SDP [18] injects task-specific expert modules into diffusion policies but requires oracle task identifiers during deployment, which prevents fully autonomous operation in continual learning scenarios. Another recent work [31] fine-tunes a pre-trained base VLA checkpoint for each new task and employs a task scheduler to select from a model library for deployment. However, this approach is memory-inefficient and does not enable knowledge sharing between tasks. MLR [32] reduces storage demands by replaying only the frozen-encoder latent embeddings of past demonstrations and regularizes task embeddings via an angular margin constraint to prevent cross-task representation collapse. Most closely related to our work, DMPEL [17] builds an expert library for a transformer-based policy [33] and iteratively fine-tunes a router network using replay of the router's past input/output data. In contrast to these works, CLARE does not require storing any previous data.

III. Problem Setup

We consider a robotic system with state $\boldsymbol{s}_t$ and action $\boldsymbol{a}_t$ at timestep t that must sequentially learn new tasks $\{\mathcal{T}_n\}_{n=1}^{N}$, with the total number of tasks, N, unknown. A task $\mathcal{T}_n = (\rho_0^n, \boldsymbol{l}_n)$ is defined by an initial distribution of the state of the robot and the environment ρ_0^n and a language instruction $\boldsymbol{l}_n$. We assume the availability of a base VLA policy $\pi_0 = \pi_{\boldsymbol{\theta}_0}$ with model parameters $\boldsymbol{\theta}_0$ that takes as input an observation $\boldsymbol{o}_t = (\boldsymbol{I}_t^1, \ldots, \boldsymbol{I}_t^{N_c}, \boldsymbol{q}_t, \boldsymbol{l})$

Algorithm 1 Continual learning for VLAs with CLARE.

Require: Pre-trained base VLA with parameters $\boldsymbol{\theta}_0$, set of expandable layers $\mathcal{E}$, expansion threshold γ.

for all layers $\ell \in \mathcal{E}$ **do**
  Set $\mathcal{A}_\ell = \emptyset$, $k_\ell = 0$. ▷ *Initialize adapter modules*
  Set $\mathcal{D}_\ell = \emptyset$. ▷ *Initialize discriminator modules*
for all tasks $\mathcal{T}_n \in \{\mathcal{T}_1, \mathcal{T}_2, \dots\}$ **do** ▷ *Continual learn.*
  Set $\boldsymbol{\theta}_n \leftarrow \boldsymbol{\theta}_{n-1}$.
  Collect demonstration data $\mathcal{D}_n$.
  for all layers $\ell \in \mathcal{E}$ **do** ▷ *Dynamic Expansion*
    for all discriminators $D_\ell^j \in \mathcal{D}_\ell$ **do**
      Compute z-score z_ℓ^j via (7).
    Expand $\mathcal{D}_\ell \leftarrow \mathcal{D}_\ell \cup \{D_\ell^n\}$. ▷ *New discriminator*
    Update model parameters: $\boldsymbol{\theta}_n \leftarrow (\boldsymbol{\theta}_n, D_\ell^n)$.
    if $n = 1$ or $z_\ell^j > \gamma$ for all $j = 1, \dots, n-1$ **then**
      Set $k_\ell \leftarrow k_\ell + 1$.
      Expand $\mathcal{A}_\ell \leftarrow \mathcal{A}_\ell \cup \{A_\ell^{k_\ell}\}$. ▷ *New adapter*
      Update model parameters: $\boldsymbol{\theta}_n \leftarrow (\boldsymbol{\theta}_n, A_\ell^{k_\ell})$.
      Link new discriminator $B_\ell(D_\ell^n) = A_\ell^{k_\ell}$.
    else
      Link D_ℓ^n to an existing adapter via (8).
  if $n > 1$ and no layers $\ell \in \mathcal{E}$ were expanded **then**
    Expand the shallowest layer $\ell_1 \in \mathcal{E}$.
    Re-link $D_{\ell_1}^n$ to the new adapter.
  Train all adapters added at stage n from $\mathcal{D}_n$ via (1).
  Train D_ℓ^n of all layers $\ell \in \mathcal{E}$ from $\mathcal{D}_n$ via (5).

consisting of camera images $\boldsymbol{I}_t^1, \dots, \boldsymbol{I}_t^{N_c}$, proprioceptive state $\boldsymbol{q}_t$ and language command $\boldsymbol{l}$, and generates an action chunk $\boldsymbol{A}_t = (\boldsymbol{a}_t, \dots, \boldsymbol{a}_{t+H-1}) \sim \pi_0(\cdot \mid \boldsymbol{o}_t)$. The first $h \leq H$ actions in $\boldsymbol{A}_t$ are applied to the robot, and the policy generates a new chunk at timestep $t+h$ in a receding horizon manner [34]. Pre-training has provided the base VLA with general visual, language, and action representations, but it cannot solve new tasks zero-shot [6], [7].

The robot should be able to learn a new task $\mathcal{T}_n$ at stage n while retaining the general knowledge from pre-training and without forgetting how to solve previous tasks $\mathcal{T}_1, \dots, \mathcal{T}_{n-1}$. Specifically, given an expert demonstration dataset $\mathcal{D}_n = \{(\boldsymbol{o}_t^n, \boldsymbol{a}_t^n), \boldsymbol{l}_n\}_{t=1}^T$ of observation-action pairs for task $\mathcal{T}_n$, we aim to train a new policy $\pi_n = \pi_{\boldsymbol{\theta}_n}$ with parameters $\boldsymbol{\theta}_n$. For the reasons discussed above, we consider exemplar-free continual learning, i.e., data from earlier stages $\mathcal{D}_1, \dots, \mathcal{D}_{n-1}$ are *not* available. Hence, only the previous model parameters $\boldsymbol{\theta}_{n-1}$ and the new data $\mathcal{D}_n$ can be used to adapt the policy to the new task.

IV. Methodology

This section details CLARE, our proposed continual learning framework for VLAs. The training and inference strategies are summarized in Algorithms 1 and 2.

A. Base Policy

To learn from high-dimensional, multimodal demonstration datasets, we train the policy using flow matching [20].

Algorithm 2 Autonomous routing during deployment.

Require: Adapters $\mathcal{A}_\ell$, discriminators $\mathcal{D}_\ell$, learned linking $B_\ell : \mathcal{D}_\ell \to \mathcal{A}_\ell$, input feature $\boldsymbol{x}_\ell$.

for all discriminators $D_\ell^j \in \mathcal{D}_\ell$ **do**
  Compute the reconstruction error $e_\ell^j(\boldsymbol{x}_\ell)$ via (4).
Select the most relevant adapter $A_\ell^* \in \mathcal{A}_\ell$ via (6).
Sum the outputs of original module and adapter via (3).

The policy at stage n learns a vector field $\boldsymbol{v}_{\boldsymbol{\theta}_n}$ that transports action-chunk samples from a simple base distribution (e.g., a Gaussian) to the target distribution. We adopt the standard conditional flow matching loss

$$\mathcal{L}(\boldsymbol{\theta}_n) = \mathbb{E}_{s,(\boldsymbol{A}^1,\boldsymbol{o}),\boldsymbol{A}^0}\left[\left\|\boldsymbol{v}_{\boldsymbol{\theta}_n}(\boldsymbol{A}^s, \boldsymbol{o}, s) - (\boldsymbol{A}^1 - \boldsymbol{A}^0)\right\|_2\right], \quad (1)$$

where $s \sim \mathcal{U}([0,1])$, $(\boldsymbol{A}^1, \boldsymbol{o}) \sim \mathcal{D}_n$, $\boldsymbol{A}^0 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and $\boldsymbol{A}^s = (1-s)\boldsymbol{A}^0 + s\boldsymbol{A}^1$. After training, we can generate new action chunks $\boldsymbol{A}_t = \boldsymbol{A}^1 \sim \pi_n(\cdot \mid \boldsymbol{o}_t)$ by Euler integration of the learned vector field, starting from Gaussian noise $\boldsymbol{A}^0 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, via $\boldsymbol{A}^{s+\delta s} = \boldsymbol{A}^s + \delta s \boldsymbol{v}_{\boldsymbol{\theta}_n}(\boldsymbol{A}^s, \boldsymbol{o}_t, s)$, with $K = 1/\delta s$ integration steps. As our method is architecture-agnostic, we keep the following sections general.

B. Modularized Adapters

The policy must continually acquire new task-specific knowledge, but it should leverage the general representations from pre-training to adapt to new tasks in a parameter-efficient way. To achieve this, we draw inspiration from the mixture-of-experts (MoE) approach in large language models (LLMs) [35], [36], which combines the outputs of specialized sub-networks during inference. However, while their number is fixed in MoE, our setup requires continually injecting parameters into the model to learn new tasks, and we aim to do this in a memory-efficient way.

Prior work [37], [38] has shown that a large fraction of factual associations and high-level knowledge in transformer-based LLMs is stored inside mid-layer feedforward network modules. Motivated by this insight, we define a set of n_e expandable modules $\mathcal{E} = \{\ell_1, \dots, \ell_{n_e}\}$ for continual learning. At each stage n, at most one adapter module is added as a side branch to each expandable layer according to the dynamic expansion strategy detailed in Section IV-D. We employ a lightweight encoder-decoder structure with ReLU activation functions for the adapters. Denoting the input feature of an expandable layer $\ell \in \mathcal{E}$ by $\boldsymbol{x}_\ell \in \mathbb{R}^{d_\ell}$, the output of the i-th adapter in that layer is given by

$$A_\ell^i(\boldsymbol{x}_\ell) = \boldsymbol{W}_{\ell,i}^{\text{up}} \text{ReLU}\left(\boldsymbol{W}_{\ell,i}^{\text{down}} \boldsymbol{x}_\ell\right), \quad (2)$$

where $\boldsymbol{W}_{\ell,i}^{\text{up}} \in \mathbb{R}^{d_\ell \times r}$, $\boldsymbol{W}_{\ell,i}^{\text{down}} \in \mathbb{R}^{r \times d_\ell}$, and $r \ll d_\ell$. We denote the set of adapters in layer ℓ at stage n by $\mathcal{A}_\ell^n = \{A_\ell^1, \dots, A_\ell^{k_\ell}\}$ with $k_\ell \leq n$. To maintain distinct representations for each task, we train only the newly added adapters on the data $\mathcal{D}_n$ and freeze the rest of the model. During inference, a routing mechanism described in Section IV-C activates one adapter $A_\ell^* \in \mathcal{A}_\ell$ per layer, and its output is

added to that of the original pre-trained module $M_\ell^{\text{pre}}(\cdot)$ as

$$M_\ell(\boldsymbol{x}_\ell) = M_\ell^{\text{pre}}(\boldsymbol{x}_\ell) + A_\ell^*(\boldsymbol{x}_\ell). \tag{3}$$

Injecting adapters as parallel side branches to the model is beneficial as it preserves the network structure and does not change the input and output of existing layers and adapters.

C. Autonomous Routing

During deployment, a routing mechanism needs to determine which adapter $A_\ell^* \in \mathcal{A}_\ell$ to activate in each layer $\ell \in \mathcal{E}$. This selection should be autonomous and based solely on the current observation, i.e., without requiring task labels, since these are typically not provided in open, real-world scenarios. Unlike fixed-size routing in MoE, our setup requires selecting from a continually increasing set of adapters. We achieve this by designing an expandable and lightweight routing mechanism that selects, for each expandable layer $\ell \in \mathcal{E}$, the adapter most relevant to the current situation, as shown in Figure 2. We pair each layer with an expanding set of autoencoder discriminators $D_\ell = \{D_\ell^1, D_\ell^2, \dots\}$, all of which receive the same features $\boldsymbol{x}_\ell$ as input, and attach a new discriminator D_ℓ^n at each stage n. Every discriminator D_ℓ^j, $j = 1, \dots, n$, is linked to one corresponding adapter $A_\ell^i = B_\ell(D_\ell^j)$ through a surjective mapping $B_\ell : D_\ell \to \mathcal{A}_\ell$, as explained in Section IV-D. We use the reconstruction errors of the discriminators

$$e_\ell^j(\boldsymbol{x}_\ell) = \left\| \boldsymbol{x}_\ell - D_\ell^j(\boldsymbol{x}_\ell) \right\|_2, \qquad j = 1, \dots, n, \tag{4}$$

to determine the most relevant adapter. By training the discriminators added at stage n with the loss

$$\mathcal{L}_{\text{recon}}(D_\ell^n) = \mathbb{E}_{\boldsymbol{x}_\ell \sim \mathcal{D}_n}\left[e_\ell^n(\boldsymbol{x}_\ell) \right], \tag{5}$$

we ensure they have a lower reconstruction error when the input features belong to the training distribution of their corresponding adapter. During inference, we activate the most relevant adapter linked to the discriminator with the smallest reconstruction error (4) via the routing mechanism

$$A_\ell^*(\boldsymbol{x}_\ell) = B_\ell\big(D_\ell^{j^*}\big), \tag{6a}$$

$$\text{where} \qquad j^* = \underset{j \in \{1,\dots,n\}}{\arg\min}\ e_\ell^j(\boldsymbol{x}_\ell). \tag{6b}$$

The routing strategy for one layer is summarized in Algorithm 2. The distribution of features $\boldsymbol{x}_\ell$ changes when training the adapters in the shallower layers. To nevertheless ensure stable discriminator training, we adopt a two-stage training strategy. First, we jointly train the new adapters using the flow-matching loss (1). Then, we freeze all parameters except for the new discriminators and train them using the reconstruction loss (5). We note that failed demonstrations, if available, could be added to the discriminators' training data to improve the robustness of the routing mechanism.

D. Dynamic Expansion

To effectively capture task-specific knowledge without catastrophic forgetting in the context of exemplar-free continual learning, some model expansion is necessary for each new task. A straightforward approach would be to add new adapters to all expandable layers. However, this limits knowledge sharing between the tasks and leads to an excessive linear increase in the number of adapter parameters. Therefore, we only expand a layer ℓ at stage n if the features of the new task $\mathcal{T}_n$ deviate substantially from all previous tasks. Since the discriminators are trained on different data, comparing their reconstruction losses requires normalization. To this end, we maintain the running mean μ_ℓ^j and standard deviation σ_ℓ^j of the reconstruction loss for each discriminator D_ℓ^j and calculate the normalized z-scores

$$z_\ell^j = \frac{1}{|\mathcal{D}_n|} \sum_{\boldsymbol{x}_\ell \in \mathcal{D}_n} \frac{e_\ell^j(\boldsymbol{x}_\ell) - \mu_\ell^j}{\sigma_\ell^j}. \tag{7}$$

If all discriminators $D_\ell^1, \dots, D_\ell^{n-1}$ have a z-score (7) larger than a threshold γ, the features $\boldsymbol{x}_\ell$ of the new task $\mathcal{T}_n$ in layer ℓ are out-of-distribution with respect to all previously learned tasks. In this case, we expand the layer by a new adapter $A_\ell^{k_\ell}$ and link the new discriminator D_ℓ^n to it, i.e., $B_\ell(D_\ell^n) = A_\ell^{k_\ell}$. Conversely, if at least one discriminator D_ℓ^j yields a z-score smaller than γ, we deem that the layer ℓ does not require expansion. This dynamic expansion strategy, illustrated in Figure 2, results in a memory-efficient, sublinear increase in the number of adapter parameters.

Even if a layer is not expanded, we need to attach a new discriminator to it. To explain why, we consider the following scenario: Assume that at stage n, a new adapter $A_{\ell_2}^i$ is added

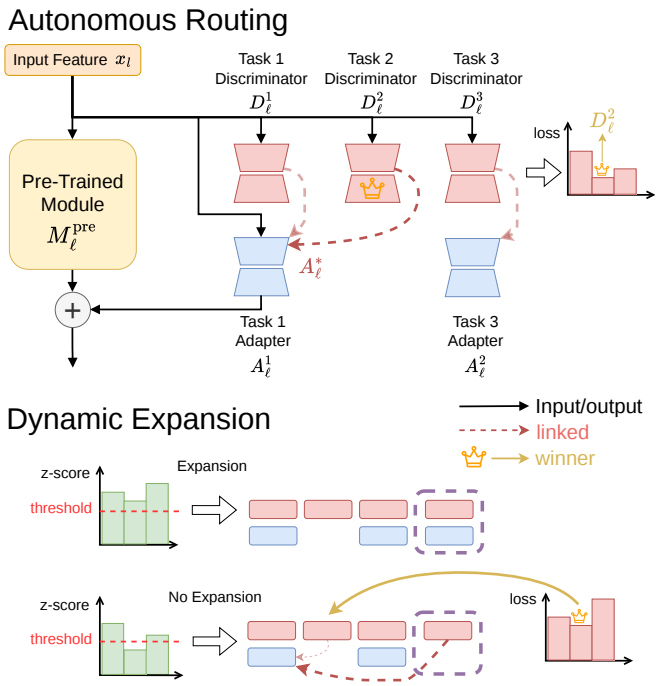


Fig. 2: CLARE sequentially adds adapters and discriminators as side branches to selected VLA modules. Top: During inference, our router activates only the most relevant adapter that is linked to the discriminator with the lowest reconstruction error for the input feature. Bottom: During dynamic expansion, if all z-scores exceed a threshold γ, a new adapter and discriminator are added to the corresponding layer. If at least one z-score is smaller than γ, we only add a discriminator and link it to the most relevant adapter.

Hyperparameter	Adapters		Discriminators	
	Real	Sim.	Real	Sim.
# Params (linear proj.)	0.38M	3.2M	6.08M	1.4M
# Params (AdaLN)	0.75M	0.26M	1.00M	0.33M
Learning rate	2×10^{-4}	1×10^{-4}	5×10^{-4}	
Learning rate schedule	cosine		constant	
Batch size	256	32	256	32
Training steps	20,000		10,000	2,000
Expansion threshold γ	0.0	2.5	–	

TABLE I: Model and training hyperparameters. The injected modules are much smaller than the base policy, which has about 200M parameters.

to layer ℓ_2, but the shallower layer ℓ_1 is not expanded. Then, the routing mechanism activates only adapters from earlier stages in layer ℓ_1 during training of $A^i_{\ell_2}$. However, a new adapter $A^j_{\ell_1}$ could be added to layer ℓ_1 at the next stage $n+1$. In this case, when revisiting task n, the router might activate the new adapter $A^j_{\ell_1}$ instead of an earlier one. As a consequence, the input features $\boldsymbol{x}_{\ell_2}$ to layer ℓ_2 when performing task $\mathcal{T}_n$ are *different* from those seen during training of $A^i_{\ell_2}$. This distribution shift in feature space can lead to unpredictable behavior and task failure [39], [40]. To avoid this problem and ensure stable routing, we attach an *auxiliary discriminator* if a layer is not expanded. The auxiliary discriminator is linked to the same adapter as the existing discriminator with the smallest reconstruction error (4), i.e.,

$$B_\ell(D^n_\ell) = A^i_\ell = B_\ell\big(D^{j^*}_\ell\big), \tag{8a}$$

$$\text{where} \qquad j^* = \underset{j\in\{1,\dots,n-1\}}{\arg\min} \; \mathbb{E}_{\boldsymbol{x}_\ell \sim \mathcal{D}_n}\big[e^j_\ell(\boldsymbol{x}_\ell)\big]. \tag{8b}$$

Intuitively, since adapter (8a) was trained on features most similar to task $\mathcal{T}_n$, we consider its learned representations to be transferable to $\mathcal{T}_n$. We note that with our dynamic expansion strategy, an adapter can potentially be activated by more than one discriminator, as shown in Figures 1 and 2.

We found that introducing at least some new parameters per task is essential for the policy to acquire and retain novel skills. In addition, we observed that shallower layers typically exhibit a stronger distribution shift between tasks than deeper layers. Thus, if no layer is deemed to require expansion, we still add an adapter to the shallowest layer $\ell_1 \in \mathcal{E}$ to capture the peculiarities of a new task. At the first stage, we expand all layers $\ell \in \mathcal{E}$ by default. In summary, our dynamic expansion mechanism ensures that CLARE adds only a small, task-dependent number of parameters without compromising performance when revisiting previous tasks.

V. Evaluation

We conduct extensive simulation and real-world experiments with a focus on the following research questions:

- **Q1:** Which layers are best suited for expansion?
- **Q2:** How well can CLARE learn new tasks, and is the performance on previous tasks affected?
- **Q3:** Can our autonomous dynamic expansion strategy reuse relevant skills from previous tasks?

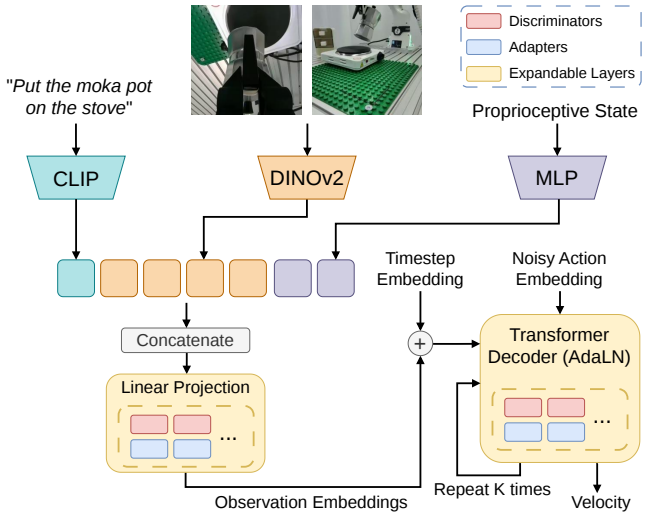


Fig. 3: Model architecture of our base VLA policy. The modules for inserting CLARE adapters are shown as dashed blocks.

- **Q4:** What is the computational overhead of CLARE?

A. Experimental Setup

1) Tasks: We conduct our simulation experiments using the LIBERO benchmark [33], which is designed specifically for continual learning. Here, a Panda manipulator with a parallel-yaw gripper needs to perform tasks in a kitchen environment, with 50 human expert demonstrations available per task. We pre-train the policy on 90 short-horizon tasks from LIBERO-90 and evaluate continual learning on 10 sequentially arriving tasks from LIBERO-Long, LIBERO-Goal, and LIBERO-Spatial. These suites require the robot to understand language instructions and execute various motions, such as pick-and-place, opening a drawer, or turning a knob.

We also conduct hardware experiments with an FR3 manipulator across five tasks, which are visualized in Figure 6:

1) Bowl: Put a bowl on a plate.
2) Stack: Stack a bowl on top of three other bowls.
3) Moka: Put a Moka pot on a stove. The pot weighs 0.5 kg and hangs at an angle when lifted by its handle.
4) Drawer: Close a drawer. The friction profile of the plastic drawer necessitates exerting considerable force.
5) Lego: Put a Lego block into the drawer and close it.

We pre-train the policy on a mix of 1000 demonstrations collected in our lab for tasks different from the five continual learning tasks and 2000 episodes from the DROID dataset [9].

2) Policy: The observations contain RGB images from wrist and third-person cameras, the end-effector pose and gripper state, and a language command. The policy generates chunks of $H = 16$ end-effector displacement actions, and the first $h = 8$ actions are sent to a Cartesian controller [41] at a frequency of 15 Hz (20 Hz in simulation). For our pre-trained base VLA, we adopt a decoder-only diffusion transformer (DiT-Dec) [42] architecture with 6 transformer

Backbone	Expandable layers	AUC ↑	FWT ↑	NBT ↓
DiT-Dec	Linear projection	**75.1**$^{\pm 1.3}$	**75.0**$^{\pm 1.4}$	**1.9**$^{\pm 0.4}$
	Decoder	41.8$^{\pm 2.4}$	45.5$^{\pm 3.8}$	7.0$^{\pm 1.7}$
DiT-EncDec	Encoder	65.4$^{\pm 2.7}$	**66.5**$^{\pm 2.2}$	1.7$^{\pm 1.2}$
	Decoder	29.0$^{\pm 2.2}$	30.9$^{\pm 4.3}$	3.0$^{\pm 3.4}$
	Encoder & Decoder	**66.6**$^{\pm 0.3}$	65.8$^{\pm 0.4}$	**1.5**$^{\pm 0.7}$

TABLE II: In our ablation study on LIBERO-Long, adding adapters to the observation encoding modules is crucial to achieve strong performance.

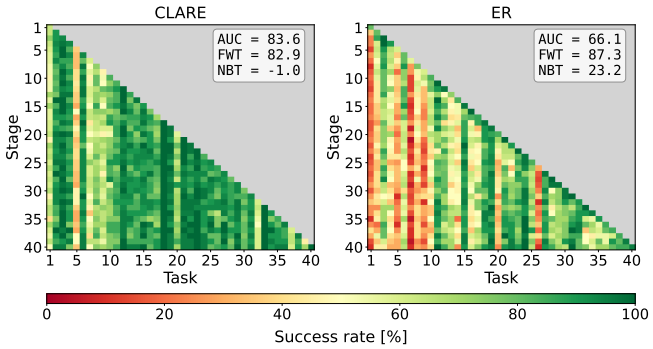


Fig. 4: CLARE scales to continual learning of 40 tasks on LIBERO-40, whereas experience replay (ER) exhibits significant performance degradation.

layers and adaptive layer normalization (AdaLN) [43] conditioning, as visualized in Figure 3. We leverage pre-trained DINOv2 [44] and CLIP [45] vision and text encoders, freezing them during continual learning. The visual and language features, as well as the proprioceptive state, are first projected into tokens of the same dimension via linear layers before being fed into the transformer backbone. Adapters can be added to the linear projection layer of the encoder and the transformer layers in the decoder. As an ablation, we also consider an encoder-decoder backbone (DiT-EncDec), for which adapters can be added to all 12 transformer layers. Our training hyperparameters are provided in Table I. Training takes about one hour per simulation task and five hours per real-world task on an NVIDIA RTX 5090 GPU.

3) Metrics: We use three metrics to assess continual learning [33], [46]: Area under the success rate curve (AUC), forward transfer (FWT), and negative backward transfer (NBT). Denoting the success rate on task n after learning the first $m \geq n$ tasks as $r_{n|m}$, the metrics are defined as $\text{AUC} = \frac{1}{N}\sum_{n=1}^{N}\left(\frac{1}{N-n+1}\sum_{m=n}^{N} r_{n|m}\right)$, $\text{FWT} = \frac{1}{N}\sum_{n=1}^{N} r_{n|n}$, and $\text{NBT} = \frac{1}{N-1}\sum_{n=1}^{N-1}\left(\frac{1}{N-n}\sum_{m=n+1}^{N}\left(r_{n|n} - r_{n|m}\right)\right)$. Intuitively, AUC measures overall performance on new and old tasks, FWT quantifies the ability to learn new tasks, and NBT measures the degree of forgetting (lower being better). All numerical results are given in percentage points.

4) Evaluation Protocol: After each stage n, we evaluate the policy on all previously learned tasks $\mathcal{T}_1, \ldots, \mathcal{T}_n$ using 100 rollouts per task (10 in the real world), with different initial configurations of relevant objects. We average all simulation results across three random seeds.

5) Baselines: We include seven baselines for continual learning without oracle task IDs. **SeqFFT** [33], [47] sequentially fine-tunes the whole model for each new task. **SeqLoRA** [28] adds LoRA adapters to selected layers, which are merged back into the base model weights after training on a new task. We add LoRA adapters to all linear layers within the projection, attention, feedforward, and AdaLN modules. **PackNet** [26] freezes the most important 25% of model weights, and the remaining free weights are used to learn the next task. This process is repeated iteratively, with each new task having fewer free weights available. **ER** [13] trains the model on a mix of 50% previous and 50% new data for each new task. **LOTUS** [29] maintains a growing skill library and uses a meta policy to compose skills during deployment. **DMPEL** [17] builds an expert library and iteratively fine-tunes the router using expert coefficient replay. **MLR** [32] replays compact joint latent representations and regularizes task embeddings to preserve inter-task distinctiveness.

B. Simulation Results

To examine the impact of the set of layers to expand $\mathcal{E}$ for LIBERO-Long, we set $\gamma = 0$, i.e., an adapter is added to each expandable layer per new task. As shown in Table II, expanding only the encoder part of the model outperforms expanding only the decoder. This indicates that the policy's observation-conditioning modules, which perform task-specific modulations of the decoder's action priors, are well-suited to store new knowledge during continual learning. Hence, we adopt this expansion strategy in the subsequent experiments.

Our baseline comparison is summarized in Table III. For DMPEL and MLR, we report the authors' results. CLARE achieves the highest overall performance, as measured by AUC, outperforming the best baseline, ER, by about 10 to 14 percentage points. Compared to SeqFFT and ER, which fine-tune the full model, CLARE achieves comparable FWT, indicating that it can store new task-specific knowledge in much fewer parameters. Moreover, our method achieves approximately zero NBT, demonstrating that it can avoid forgetting without relying on exemplar data or oracle task identifiers.

We examine long-term scalability on LIBERO-40, which concatenates all four standard LIBERO suites (LIBERO-Long, -Goal, -Spatial, -Object). Due to computational constraints, we perform only 10 rollouts per task and stage across three seeds. As shown in Figure 4, CLARE can sequentially learn and retain 40 distinct tasks, demonstrating the scalability and robustness of our autonomous routing strategy. In contrast, ER cannot avoid catastrophic forgetting of several tasks (e.g., $\mathcal{T}_1$ and $\mathcal{T}_7$), yielding an NBT of 23%.

We evaluate the impact of the expansion threshold γ for DiT-EncDec with expandable encoder layers and report the results in Figure 5. Increasing γ from 0 to 20 reduces the number of added adapters from 60 to 16, resulting in a slight decrease in AUC and FWT. Intuitively, compressing new knowledge into fewer model parameters reduces the robot's ability to learn novel tasks. Yet, AUC remains higher than

Method	LIBERO-Long			LIBERO-Goal			LIBERO-Spatial		
	AUC ↑	FWT ↑	NBT ↓	AUC ↑	FWT ↑	NBT ↓	AUC ↑	FWT ↑	NBT ↓
SeqFFT	$22.4^{\pm 0.3}$	$\underline{76.1}^{\pm 1.0}$	$74.7^{\pm 1.1}$	$26.7^{\pm 0.9}$	$\underline{94.1}^{\pm 0.2}$	$95.3^{\pm 1.4}$	$27.7^{\pm 0.6}$	$\mathbf{94.7}^{\pm 0.3}$	$94.6^{\pm 0.9}$
SeqLoRA	$21.4^{\pm 1.0}$	$73.1^{\pm 1.8}$	$71.6^{\pm 1.6}$	$26.1^{\pm 0.3}$	$90.1^{\pm 1.6}$	$90.8^{\pm 1.7}$	$27.3^{\pm 1.3}$	$90.1^{\pm 2.1}$	$89.2^{\pm 1.3}$
PackNet	$4.8^{\pm 0.2}$	$37.2^{\pm 1.0}$	$41.3^{\pm 1.2}$	$10.5^{\pm 0.3}$	$60.3^{\pm 1.0}$	$67.0^{\pm 1.1}$	$8.6^{\pm 0.1}$	$54.7^{\pm 0.5}$	$60.3^{\pm 0.7}$
ER	$\underline{60.5}^{\pm 0.2}$	$\mathbf{76.6}^{\pm 0.9}$	$22.7^{\pm 1.8}$	$76.0^{\pm 0.9}$	$\mathbf{94.4}^{\pm 0.7}$	$25.1^{\pm 0.5}$	$\underline{77.6}^{\pm 0.8}$	$\underline{92.7}^{\pm 1.0}$	$20.9^{\pm 2.0}$
LOTUS	$52.9^{\pm 1.6}$	$58.1^{\pm 0.2}$	$\mathbf{-7.2}^{\pm 3.0}$	$56.0^{\pm 1.0}$	$61.0^{\pm 3.0}$	$30.0^{\pm 1.0}$	NA	NA	NA
DMPEL	$58^{\pm 3}$	$55^{\pm 4}$	$7^{\pm 1}$	$\underline{78}^{\pm 2}$	$68^{\pm 2}$	$\mathbf{0}^{\pm 1}$	$70^{\pm 3}$	$64^{\pm 2}$	$\underline{3}^{\pm 1}$
MLR	NA	NA	NA	$77.2^{\pm 1.8}$	$80.0^{\pm 2.5}$	$6.9^{\pm 0.9}$	NA	NA	NA
CLARE (ours)	$\mathbf{75.1}^{\pm 1.3}$	$75.0^{\pm 1.4}$	$\underline{1.9}^{\pm 0.4}$	$\mathbf{89.3}^{\pm 1.1}$	$89.7^{\pm 1.5}$	$\mathbf{0.3}^{\pm 1.1}$	$\mathbf{87.4}^{\pm 2.3}$	$88.0^{\pm 1.9}$	$\mathbf{0.9}^{\pm 0.6}$

TABLE III: Baseline comparison across three LIBERO suites. CLARE achieves the highest overall performance, as measured by AUC, and demonstrates strong capabilities to acquire new skills without forgetting. "NA" indicates not available.

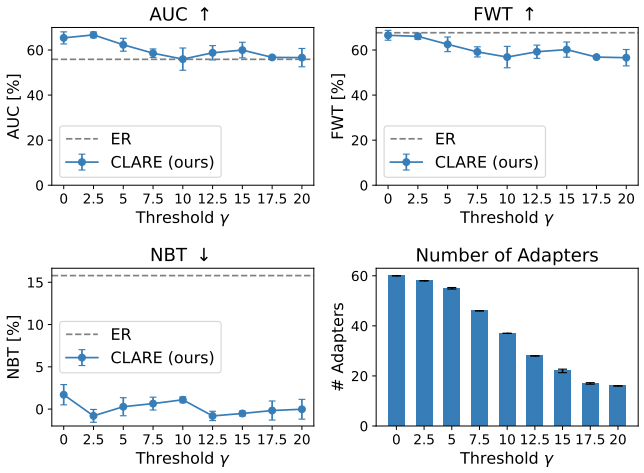


Fig. 5: Increasing the dynamic expansion threshold γ reduces the number of added adapters and, consequently, the capability to learn new tasks (lower FWT), but does not lead to catastrophic forgetting (approximately zero NBT).

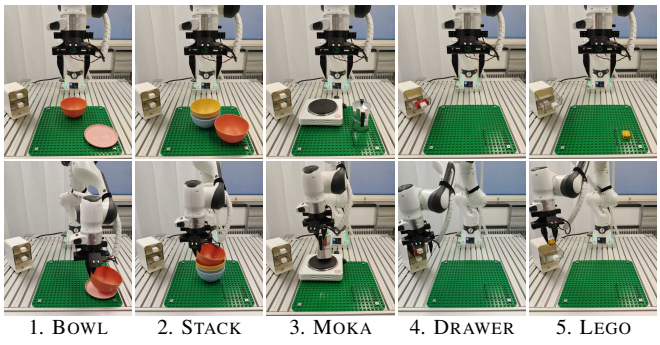


Fig. 6: Our five real-world manipulation tasks involve objects of different shapes, weights, and dynamics, as well as different motion patterns.

for ER, and NBT stays close to zero, indicating that the policy does not exhibit forgetting even when only a few adapters are added per task. Comparing the results in Table II and Figure 5, we find that the choice of expandable layers can have a stronger impact than the expansion threshold value. In summary, γ should be chosen based on the importance of high task performance versus low memory requirements.

Method	AUC ↑	FWT ↑	NBT ↓
SeqFFT	23.8	**68.0**	80.0
SeqLoRA	22.9	64.0	76.9
ER	51.1	60.0	17.1
CLARE (ours)	**63.3**	62.0	**-2.9**

TABLE IV: Overall results in our hardware experiments.

Stage	1. Bowl ①	1. Bowl ②	2. Stack ①	2. Stack ②	3. Moka ①	3. Moka ②	4. Drawer ①	4. Drawer ②	5. Lego ①	5. Lego ②
1	**100**	**100**	–	–	–	–	–	–	–	–
2	**100**	80	**70**	50	–	–	–	–	–	–
3	**80**	**80**	50	**60**	**20**	**20**	–	–	–	–
4	80	**90**	**70**	50	20	**50**	80	**90**	–	–
5	60	**90**	50	**60**	10	**30**	50	**90**	30	**50**

TABLE V: Evolution of per-task success rates [%] across all stages in our real-world experiments. ① ER, ② CLARE.

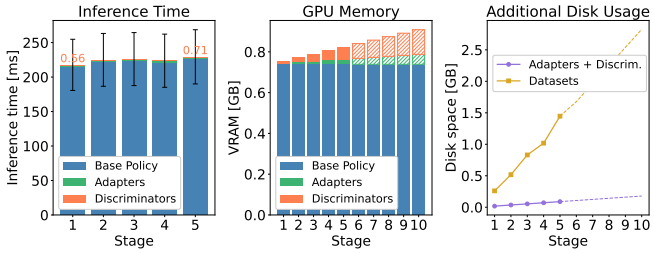


Fig. 7: Inference time and memory complexity of CLARE in our hardware experiments. The values for stages 6-10 are linearly extrapolated.

C. Real-World Results

Given the importance of expanding the observation encoding modules (see Table II), we add adapters to the linear projection layers and the scaling and shifting AdaLN modules of each transformer decoder layer for our hardware experiments. To keep inference latency and memory overhead low, we implement the adapters using LoRA. Due to the scale of the hardware evaluation, we focus on three baselines: SeqFFT, SeqLoRA, and ER. The results are provided in Tables IV and V. CLARE achieves an AUC of 63%, which is 12 percentage points higher than ER, and shows no catastrophic forgetting with an NBT of -2.9%. SeqFFT and SeqLoRA achieve high performance on new tasks, but cannot sufficiently retain the relevant representations from previous tasks. In summary, CLARE demonstrates strong capabilities for continual learning under real-world operating conditions.

Figure 7 analyzes the inference time and memory overhead of CLARE. The inference-time overhead relative to the base policy is below 3 ms, and GPU VRAM utilization increases only by about 2% per task. Compared to explicitly storing data from previous tasks for ER, our approach of acquiring new skills by training dedicated lightweight modules is more storage-efficient and reduces forgetting to near zero in our experiments.

VI. Conclusions

CLARE enables continual learning without forgetting in VLAs, requiring neither stored exemplars nor task IDs. By combining lightweight adapters, an autonomous expansion strategy, and an autoencoder-based routing module, our approach increases model capacity only when needed while retaining prior representations. Across multiple LIBERO suites, CLARE achieves and maintains high task success, outperforming even strong baselines that have access to previous data. Hardware experiments on five manipulation tasks with diverse interaction dynamics confirm these findings, demonstrating the potential of CLARE for long-term deployment of robots in non-stationary, real-world environments.

Acknowledgements

This work was supported by the German Federal Ministry of Research, Technology and Space (BMFTR) under the Robotics Institute Germany (RIG) funded by BMFTR grant 16ME0997K, the German Research Foundation (DFG) within the RTG project ConVeY funded by grant GRK 2428, and the Humboldt Professorship for Robotics and Artificial Intelligence.

References

[1] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[2] S. Thrun and T. M. Mitchell, "Lifelong robot learning," *Robotics and Autonomous Systems*, vol. 15, no. 1, pp. 25–46, 1995.

[3] A. Billard *et al.*, "A roadmap for AI in robotics," *Nature Machine Intelligence*, vol. 7, no. 6, pp. 818–824, 2025.

[4] S. Dohare, J. F. Hernandez-Garcia, Q. Lan, P. Rahman, A. R. Mahmood, and R. S. Sutton, "Loss of plasticity in deep continual learning," *Nature*, vol. 632, no. 8026, pp. 768–774, 2024.

[5] M. J. Kim *et al.*, "OpenVLA: An open-source vision-language-action model," in *Conference on Robot Learning*, 2025, pp. 2679–2713.

[6] P. Intelligence *et al.*, "$\pi_{0.5}$: A vision-language-action model with open-world generalization," *Conference on Robot Learning*, 2025.

[7] M. Reuss, H. Zhou, M. Rühle, Ö. E. Yağmurlu, F. Otto, and R. Lioutikov, "FLOWER: Democratizing generalist robot policies with efficient vision-language-action flow policies," in *Conference on Robot Learning*, 2025.

[8] M. Shukor *et al.*, "SmolVLA: A vision-language-action model for affordable and efficient robotics," *arXiv preprint arXiv:2506.01844*, 2025.

[9] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis *et al.*, "DROID: A large-scale in-the-wild robot manipulation dataset," in *Robotics: Science and Systems*, 2024.

[10] A. O'Neill *et al.*, "Open X-embodiment: Robotic learning datasets and RT-X models," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2024, pp. 6892–6903.

[11] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in Cognitive Sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[12] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," *Advances in Neural Information Processing Systems*, 2017.

[13] A. Chaudhry *et al.*, "On tiny episodic memories in continual learning," *arXiv preprint arXiv:1902.10486*, 2019.

[14] A. Xie and C. Finn, "Lifelong robotic reinforcement learning by retaining experiences," in *Conference on Lifelong Learning Agents*. PMLR, 2022, pp. 838–855.

[15] Z. Liu *et al.*, "TAIL: Task-specific adapters for imitation learning with large pretrained models," *International Conference on Learning Representations*, 2024.

[16] H. Wang, H. Lu, L. Yao, and D. Gong, "Self-expansion of pre-trained models with mixture of adapters for continual learning," in *Proceedings of the Computer Vision and Pattern Recognition Conference*, 2025, pp. 10 087–10 098.

[17] Y. Lei, S. Mao, S. Zhou, H. Zhang, X. Li, and P. Luo, "Dynamic mixture of progressive parameter-efficient expert library for lifelong robot learning," *arXiv preprint arXiv:2506.05985*, 2025.

[18] Y. Wang *et al.*, "Sparse diffusion policy: A sparse, reusable, and flexible policy for robot learning," in *Conference on Robot Learning*, 2024.

[19] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in Neural Information Processing Systems*, pp. 6840–6851, 2020.

[20] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *International Conference on Learning Representations*, 2023.

[21] X. Chen *et al.*, "PaLI-3 vision language models: Smaller, faster, stronger," *arXiv preprint arXiv:2310.09199*, 2023.

[22] F. Lin, Y. Hu, P. Sheng, C. Wen, J. You, and Y. Gao, "Data scaling laws in imitation learning for robotic manipulation," in *International Conference on Learning Representations*, 2025.

[23] D.-W. Zhou *et al.*, "Learning without forgetting for vision-language models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 47, no. 6, pp. 4489–4504, 2025.

[24] J. Kirkpatrick *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[25] F. Zenke, B. Poole, and S. Ganguli, "Continual learning through synaptic intelligence," in *International Conference on Machine Learning*, 2017, pp. 3987–3995.

[26] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.

[27] A. A. Rusu *et al.*, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[28] E. J. Hu *et al.*, "LoRA: Low-rank adaptation of large language models." in *International Conference on Learning Representations*, 2022.

[29] W. Wan, Y. Zhu, R. Shah, and Y. Zhu, "LOTUS: Continual imitation learning for robot manipulation through unsupervised skill discovery," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2024, pp. 537–544.

[30] Y. Zhu, P. Stone, and Y. Zhu, "Bottom-up skill discovery from unsegmented demonstrations for long-horizon robot manipulation," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4126–4133, 2022.

[31] L. Xie, Y. Li, and H. Zhuang, "Analytic task scheduler: Recursive least squares based method for continual learning in embodied foundation models," *arXiv preprint arXiv:2506.09623*, 2025.

[32] F. Yu, M. Tiezzi, T. Apicella, C. Beyan, and V. Murino, "Lifelong imitation learning with multimodal latent replay and incremental adjustment," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

[33] B. Liu *et al.*, "LIBERO: Benchmarking knowledge transfer for lifelong robot learning," *Advances in Neural Information Processing Systems*, pp. 44 776–44 791, 2023.

[34] C. Chi *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," in *Robotics: Science and Systems*, 2023.

[35] N. Shazeer *et al.*, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," in *International Conference on Learning Representations*, 2017.

[36] D. Dai *et al.*, "DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models," in *Annual Meeting of the*

Association for Computational Linguistics (Volume 1: Long Papers), 2024, pp. 1280–1297.
[37] K. Meng, D. Bau, A. Andonian, and Y. Belinkov, "Locating and editing factual associations in gpt," *Advances in Neural Information Processing Systems*, pp. 17 359–17 372, 2022.
[38] M. Geva, R. Schuster, J. Berant, and O. Levy, "Transformer feed-forward layers are key-value memories," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2021, pp. 5484–5495.
[39] Q. Gu *et al.*, "SAFE: Multitask failure detection for vision-language-action models," *Advances in Neural Information Processing Systems*, 2025.
[40] R. Römer, A. Kobras, L. Worbis, and A. P. Schoellig, "Failure prediction at runtime for generative robot policies," *Advances in Neural Information Processing Systems*, 2025.
[41] D. S. J. Pro, O. Hausdörfer, R. Römer, M. Dösch, M. Schuck, and A. P. Schoellig, "Crisp-compliant ros2 controllers for learning-based manipulation policies and teleoperation," *IEEE Robotics and Automation Practice*, 2026.
[42] S. Dasari, O. Mees, S. Zhao, M. K. Srirama, and S. Levine, "The ingredients for robotic diffusion transformers," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2025.
[43] W. Peebles and S. Xie, "Scalable diffusion models with transformers," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4195–4205.
[44] M. Oquab *et al.*, "DINOv2: Learning robust visual features without supervision," *Transact. on Machine Learning Research*, 2024.
[45] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 8748–8763.
[46] N. Díaz-Rodríguez *et al.*, "Don't forget, there is more than forgetting: New metrics for continual learning," in *Continual Learning Workshop at NeurIPS 2018*, 2018, pp. 1–7.
[47] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback," *Advances in Neural Information Processing Systems*, pp. 27 730–27 744, 2022.

VII. Appendix

A. Details about the Experimental Setup

Our five real-world tasks shown in Figure 6 introduce multiple levels of diversity:

- *Different objects*: The tasks involve bowls, a plate, a moka pot, a stove, a drawer, and Lego blocks. Besides visual differences, these objects also have very different weights. For example, the moka pot weighs $0.5\,\mathrm{kg}$, about 70 times as much as the Lego block.
- *Grasp configurations*: The tasks require different gripper widths (e.g., fully closed for the bowls and about half-closed for the Lego block) and angles (vertical for the Lego block and at a 30° angle to close the drawer) for object manipulation.
- *Dynamic differences*: The tasks involve distinct interaction dynamics between the robot and its environment. For example, in Moka, the moka pot hangs at an angle when lifted by its handle, and the robot must use friction between the pot and the stove to place it upright. In Drawer and Lego, the plastic drawer compartment exhibits a strongly nonlinear friction profile, requiring the robot to exert significant force to close the drawer.
- *Strategy switching*: Lego, for example, consists of three stages: picking up the block, placing it in the drawer, and closing the drawer.

B. Additional Experiments

Figure 8 shows the continual learning behavior on LIBERO-Long for a selected set of baselines. These results further demonstrate that CLARE does not exhibit catastrophic forgetting while achieving a high overall success rate across tasks.

Stage	1. Bowl ③	1. Bowl ⑥	2. Stack ③	2. Stack ⑥	3. Moka ③	3. Moka ⑥	4. Drawer ③	4. Drawer ⑥	5. Lego ③	5. Lego ⑥
1	90	**100**	–	–	–	–	–	–	–	–
2	**100**	80	20	**50**	–	–	–	–	–	–
3	**100**	80	20	**60**	**20**	**20**	–	–	–	–
4	**90**	**90**	40	**50**	40	**50**	60	**90**	–	–
5	**100**	90	**60**	**60**	20	**30**	60	**90**	30	**50**

TABLE VI: Evolution of per-task success rates [%] across all stages in our real-world experiments for different numbers of expandable AdaLN modules. ③ three modules, ⑥ six modules.

In another experiment, we investigate how quickly and reliably the router responds to *changes* in the active task during deployment. To this end, we change the language command from *"put the Lego block into the drawer"* to *"close the drawer"* mid-execution after three seconds. The router immediately switches to the correct adapter; the robot stops approaching the block and moves directly toward the drawer. An exemplary rollout is shown in Figure 9. Crucially, tasks Drawer and Lego involve almost identical scenes (same workbench, same drawer), so the router must rely on subtle differences in the language and proprioceptive features to distinguish them. This behavior was consistent across 10 rollouts, demonstrating that our reconstruction-based routing can robustly switch capabilities to adapt to dynamic context changes during deployment. We further test the router in a continuous multi-task scenario: by changing the language prompt and rearranging the scene between tasks, CLARE successfully completes a sequence of three different tasks (Lego → Bowl → Drawer) without any task identifiers, activating the correct adapter for each context.

We also evaluate the impact of the number of expandable AdaLN modules in our hardware experiments and provide the results in Table VI. Injecting adapters into all scale and shift modules performs best.

C. Extended Discussion

In our ablation experiment (see Table II), expanding only the encoder performs similarly to expanding both the encoder and the decoder, but much better than expanding only the decoder. Our main conclusion from these results is that the observation-conditioning modules are crucial for injecting adapters during continual learning.

Based on these findings, we expand only the modules in the conditioning pathway of the DiT-Dec architecture for our real-world experiments: 1) the linear layers projecting the observation features into the same token dimension, and 2) the scale and shift AdaLN modules, which inject the observation condition into the transformer decoder layers. Our hardware experiments involve changes in daylight, reflections, and slight drifts of the camera extrinsics.

Despite the physical diversity of the real-world tasks, CLARE achieves an AUC of 63.3% and a near-zero NBT of -2.9% in our hardware experiments (see Table IV),

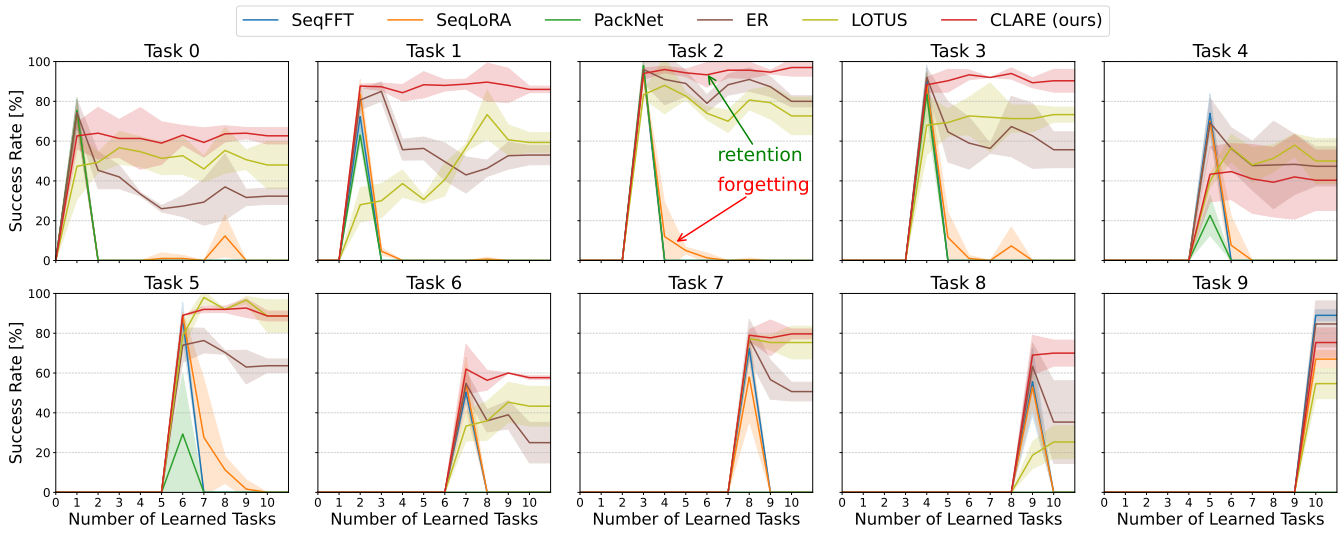
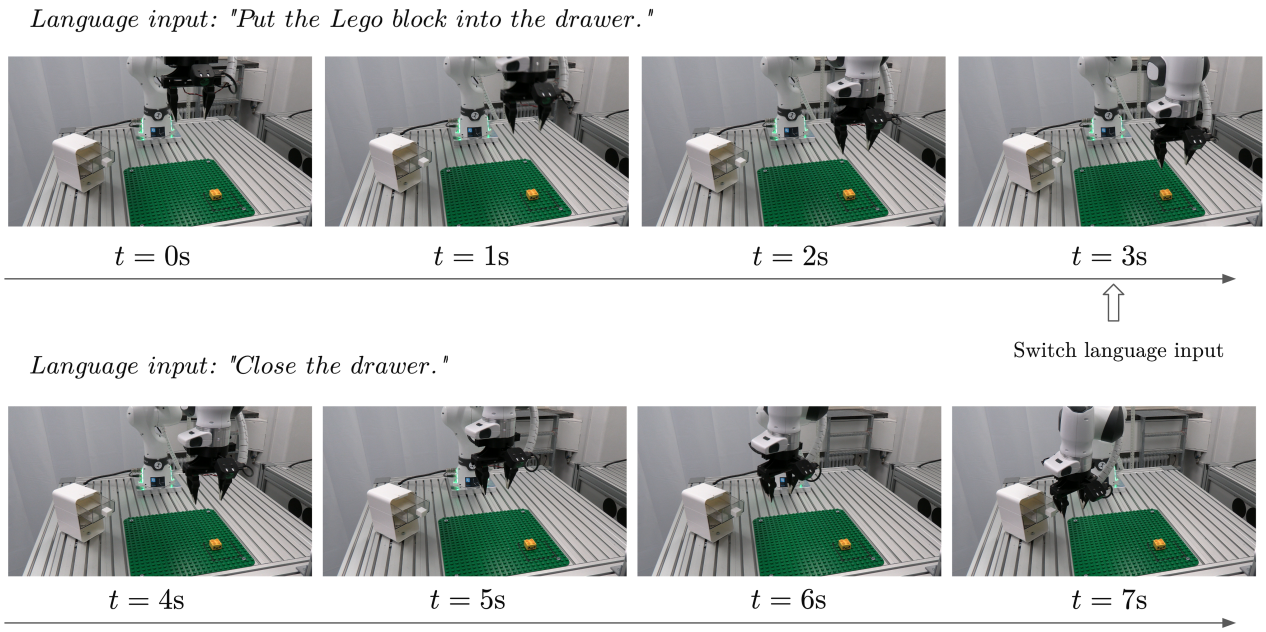


Fig. 8: Success rate curves of CLARE and five baselines on the LIBERO-Long benchmark. The solid lines represent the average success rates across three random seeds, and the shaded regions indicate the standard deviations. The results demonstrate that our method achieves a higher overall success rate and mitigates catastrophic forgetting more effectively during continual learning than the baselines, despite ER and LOTUS using previous data.

significantly outperforming all baselines. The detailed per-stage success rates in Table V further confirm that our expansion strategy preserves performance on earlier tasks even as new, physically distinct tasks are learned. These results empirically validate that focusing the expansion on the observation-conditioning pathway is not a bottleneck for physical dynamics in our setup, supporting the conclusion drawn from the simulation ablation. However, we note that this expansion strategy assumes that the VLA has been pre-trained on a sufficiently large dataset of robot demonstrations that cover diverse motion patterns. With the availability of large-scale robotic datasets, such as DROID [9], we believe this prerequisite can be met for common robot embodiments.



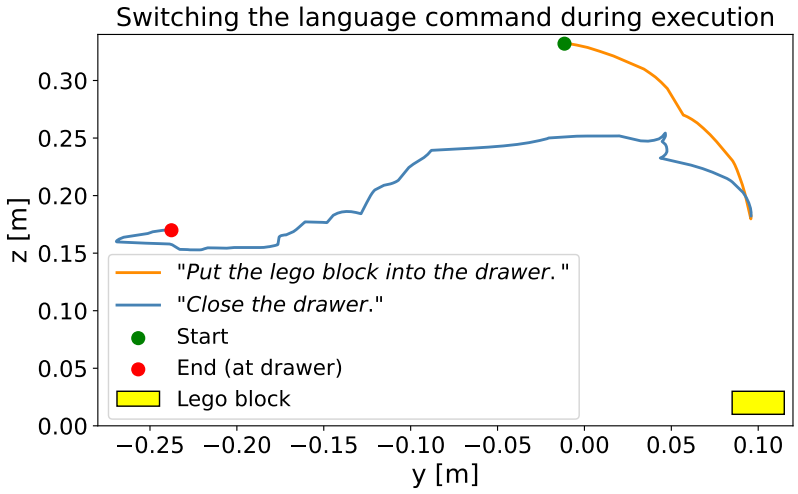


Fig. 9: When changing the language command during execution, CLARE correctly adapts its behavior by switching the activated adapters.