Continually Evolving Skill Knowledge in Vision Language Action Model

Yuxuan Wu[1,2], **Guangming Wang**[3], **Zhiheng Yang**[1], **Tianchen Deng**[1,4,5], **Maoqing Yao**[6], **Brian Sheil**[3], **Hesheng Wang**[1*]

[1]Shanghai Jiao Tong University [2]Shanghai Innovation Institute [3]University of Cambridge
[4]Nanyang Technological University [5]MIT SMART [6]AgiBot

{furrygreen, zhiheng.yang, wanghesheng}@sjtu.edu.cn {gw462, bbs24}@cam.ac.uk

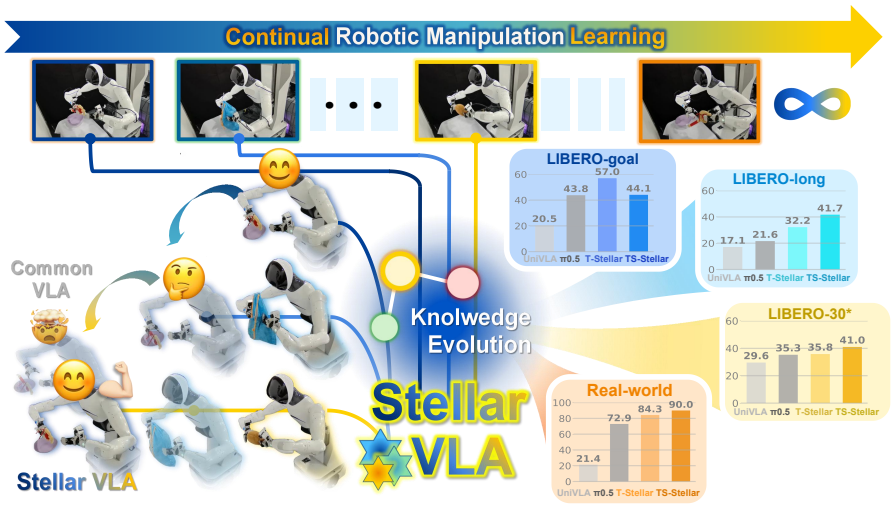


Figure 1: Illustration of Stellar VLA. T-Stellar and TS-Stellar outperform UniVLA [1] and $\pi_{0.5}$ [2] in final average success rates across simulation and real-world dual-arm experiments.

Abstract

Vision-language-action (VLA) models show promising knowledge accumulation ability through pretraining, yet continual learning remains challenging. Existing continual imitation learning (CIL) methods often require additional parameters or modules, limiting scalability for large VLA models. We propose Stellar VLA, a knowledge-driven CIL framework without parameter growth, featuring two variants: T-Stellar for task-centric modeling and TS-Stellar for hierarchical modeling of reusable skills across tasks. Stellar VLA jointly optimizes task representations and a self-evolving knowledge space, with knowledge-posterior-guided expert routing using knowledge relations and Top-K semantic embeddings for task specialization. Experiments on LIBERO demonstrate strong performance against VLA and CIL baselines with only 1% replay. Dual-arm real-world experiments validate knowledge transfer across embodiments and scenes. TS-Stellar excels in hierarchical manipulation, while visualizations demonstrate knowledge retention and task discovery. Project Page: `https://stellarvla.github.io/`.

*Corresponding author. The authors affiliated with Shanghai Jiao Tong University (Yuxuan Wu, Zhiheng Yang, Tianchen Deng, and Hesheng Wang) are with the School of Automation and Intelligent Sensing, Shanghai Jiao Tong University, and the Shanghai Key Laboratory of Navigation and Location Based Services, Shanghai 200240, China.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

A core requirement for lifelong agents is continual learning that preserves prior knowledge while adapting to new situations. Prior work on robotic continual imitation learning (CIL) addresses catastrophic forgetting through skill expansion [3] or parameter-efficient adaptation [4, 5], demonstrating promising performance in small- to medium-scale policy models. Recent advances in VLA models [1, 6, 2] demonstrate strong task learning capabilities in accumulating task experience from large-scale robotic data. However, adapting conventional CIL methods to VLAs introduces significant training and storage overhead, due to their reliance on increasing adapters or task-specific modules.

Encouragingly, recent work [7] shows that VLA models can achieve strong continual learning performance with simple experience replay [8] strategy. While promising, this result also challenges the necessity of increasingly complex CIL algorithmic designs. At this intersection between VLA scaling and conventional CIL methods, we seek to further investigate whether VLA models can reduce both the dependence on large-scale pretraining data and data replay during continual learning, thereby lowering training cost and storage overhead toward more scalable lifelong learning.

We argue that effective continual learning in VLA models should focus on enabling the model to capture task and skill relationships from data. Recent task-centric representation learning methods [9, 10, 11] have demonstrated improved policy adaptation, while LEGION [12] extends this paradigm to state-based continual reinforcement learning for incremental skill acquisition. Meanwhile, Mixture-of-Experts (MoE) architectures [13] enable dynamic routing across specialized experts, offering a mechanism to mitigate task interference. These insights motivate a key question: *Can VLA models learn latent task knowledge for continual policy evolution via adaptive specialization?*

To explore this question, we propose Stellar VLA, an end-to-end framework for CIL in vision-language-based manipulation. Building upon Dirichlet-process-based knowledge modeling and alternating optimization [12], we develop two variants for continual task and skill acquisition. T-Stellar employs a Dirichlet Process Mixture Model (DPMM) for task-centric modeling, while TS-Stellar extends it with a Hierarchical Dirichlet Process (HDP) to capture reusable skills across tasks. Alternating updates between task representations and knowledge distributions enable their co-evolution. We further integrate the learned knowledge into the policy through task-knowledge-guided expert routing, conditioned on knowledge relation embeddings and Top-K semantic embeddings. This design balances parameter sharing and specialization without increasing model size.

Stellar VLAs are evaluated in both simulation and real-world settings. On the LIBERO benchmark, T-Stellar and TS-Stellar outperform VLA [14, 1, 6, 2] and CIL [8, 15, 3, 5] baselines under both scratch and pretrained settings with only 1% data replay, demonstrating strong continual learning performance. Real-world dual-arm experiments further show effective cross-embodiment adaptation, with TS-Stellar excelling in modeling complex hierarchical actions. Visualizations demonstrate skill acquisition while retaining prior knowledge. To conclude our contributions:

- We present **Stellar VLA**, a CIL framework that models a Dirichlet Process–based knowledge space and jointly optimizes task representations and knowledge distribution via a self-evolving learning scheme. Two variants are introduced: **T-Stellar** using DPMM for Task-centric modeling, and **TS-Stellar** extending to HDP for hierarchical Task–Skill structure.
- We propose knowledge-posterior-guided expert routing using knowledge relation and Top-K semantic embeddings, enabling efficient specialization without increasing model size.
- Experiments on LIBERO and real-world tasks demonstrate robust improvements in task knowledge adaptation, with consistent gains across different embodiments, while both variants capture evolving task or skill structure.

2 Related Work

Vision-Language-Action Models. Vision-language-action (VLA) models learn unified policies by jointly modeling perception, language, and control from robotic data. Recent works such as RT-2 [18] and OpenVLA [19] demonstrate strong cross-task generalization, and more recent advances including π_0 and $\pi_{0.5}$ [6, 2], further improve scalability and efficiency, where $\pi_{0.5}$ additionally leverages auxiliary multimodal data for representation learning. These advances make VLA a promising paradigm for general-purpose robot learning, but also raise challenges for continual adaptation.

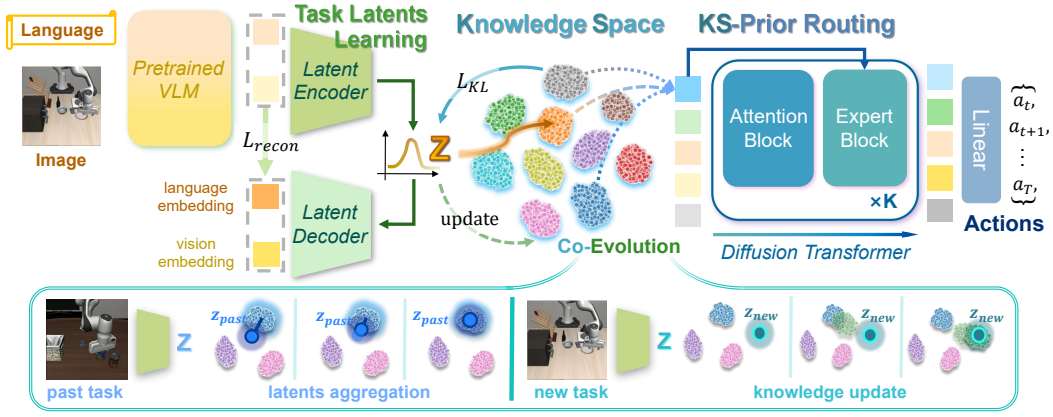


Figure 2: **Overall architecture of Stellar VLA.** CLIP [16] and FiLM [17]-conditioned ResNet encode language and visual inputs respectively. Task-centric representation z and knowledge space are jointly learned through knowledge update and latents aggregation, as detailed in Section 3.3. The learned knowledge prior guides the MoE action head for motion prediction, as detailed in Section 3.4.

Continual Learning in Robotics. To address catastrophic forgetting in robotic policy learning, continual imitation learning (CIL) has been widely studied. Early methods are typically based on experience replay [8] and parameter isolation [20]. Recent work focuses on structured policy learning, including hierarchical skill expansion [3, 21, 22] and parameter-efficient adaptation [4, 5, 23, 24, 25]. However, these methods typically introduce additional data processing requirements or model parameters, leading to higher data and training costs when scaling to VLA models, which often include over 1B parameters and require large-scale training.

Recent work [26, 7] shows that pretrained VLA models [6] can achieve strong CIL performance with simple replay using only 20% data or latent space rehearsal. This suggests promising scalability, yet reducing storage while improving task knowledge adaptation remains an open challenge. We address this by proposing a fixed-size VLA model that enables efficient CIL with only 1% data replay.

Task-centric Representation Learning. Learning task-centric representations has become a promising direction for improving cross-task learning. Existing approaches, mainly studied in multi-task settings, model task structure through compact action representations [27, 9, 11] or vision-language abstractions [28, 29, 30, 31, 32, 33], with UniVLA [1] exemplifying the latter.

Recent work further explores more flexible structures for task adaptation, including grounded task expansion [34] and action-centric incremental modeling [35, 36], but often relies on manual annotations or parameter expansion. LEGION [12] introduces a non-parametric knowledge space for state-based lifelong reinforcement learning. Inspired by LEGION, we adopt a self-evolving task knowledge space for vision-language continual imitation learning and extend it with hierarchical task–skill modeling and knowledge-guided expert routing.

3 Method

In this section, we present the architecture of Stellar VLA (Figure 2), which enables CIL for vision-language manipulation. Our goal is to continually evolve task-relevant knowledge, enabling Stellar VLA to acquire new skills while mitigating forgetting of prior tasks. We first define the CIL setting, then describe the design and evolution of the knowledge space, and finally explain the process of guiding the expert policy via knowledge priors.

3.1 Problem Formulation

In the CIL setting, an agent learns sequentially from a stream of tasks $\{\mathcal{T}_j\}_{j=1}^{\infty}$, where each task j provides N expert demonstrations $\{\tau_j^n\}_{n=1}^{N}$, consisting of language instructions $\boldsymbol{l}$, observations $\boldsymbol{o}$, and actions $\boldsymbol{a}$. Unlike multi-task learning assuming full access to all demonstrations, CIL requires

acquiring new task knowledge under limited access to past data while retaining previously learned skills, reflecting the realistic demands of lifelong robot learning.

To alleviate forgetting caused by limited access to past tasks, we adopt Experience Replay (ER) with a small buffer $\mathcal{B}$. After completing task j, $q\%$ of its demonstrations are stored in $\mathcal{B}$ and jointly trained with current data when learning task $j{+}1$. However, relying on only a small portion of past data often leads to prediction drift on previous tasks $\mathcal{T}_k$ ($k < j{+}1$). To address this, we design a VLA architecture that continually discovers and preserves task knowledge, enabling dynamic expert composition for better parameter sharing and reduced interference.

3.2 Dirichlet-Process-based Knowledge Space

To capture structured and interpretable task knowledge, we go beyond task latents and organize them into a high-level, dynamically evolving knowledge space. While latent features encode task-relevant variations, they lack explicit task identity, making it difficult to distinguish tasks during continual learning. To model such an evolving space, we adopt a Dirichlet Process (DP) prior, defined as $G \sim DP(\alpha, G_0)$, which supports clustering with an unbounded number of components. Here, G_0 denotes the base distribution over latent knowledge components, and G defines a stochastic measure over task or skill clusters. Samples from G induce clustering via shared components while allowing new components to emerge as tasks evolve (details in Appendix B). Building on Dirichlet Process, we design two knowledge spaces: a task-centric space and a hierarchical task-skill space.

Task-Centric Knowledge Modeling from DPMM. Following LEGION's task-level knowledge modeling [12], we instantiate a Dirichlet Process Mixture Model (DPMM) to model task-level structure in the latent space. Based on Dirichlet-Process sampling, task latent z_j in task j under the task-set probability measure G can be expressed as $z_j \sim F_{task}(\theta_j), \theta_j \sim G$, where θ_j denotes task-specific distribution parameters and F_{task} is the observation model. Assuming Gaussian components, this formulation induces a dynamic clustering of task representations in the knowledge space.

Hierarchical Task-Skill Knowledge Modeling from HDP. The DPMM-based formulation models tasks via cluster assignments under a shared global mixture, but robotic tasks often share reusable subskills. To address this, we adopt a Hierarchical Dirichlet Process (HDP) [37] to model shared skill components across tasks. Specifically, each task j is modeled as a distribution over skills indexed by i, where skill-level latent variables z_{ji} are generated via:

$$z_{ji} \sim F_{skill}(\theta_{ji}), \quad \theta_{ji} \sim G_j, \quad G_j \sim DP(\gamma, G), \quad G \sim DP(\alpha, G_0) \tag{1}$$

Here, z_{ji} denotes the latent representation of skill i in task j, θ_{ji} is its corresponding distribution parameter, and G_j is the task-specific measure drawn from skill-set measure G. Under diagonal Gaussian assumptions, each skill is parameterized as $\theta_i^{skill} = (\mu_i^{skill}, \sigma_i^{skill})$. Given a task j, its overall distribution parameter $\theta_j^{task} = (\mu_j^{task}, \sigma_j^{task})$ is obtained by aggregating its skill components:

$$\boldsymbol{\mu}_j^{\text{task}} = \sum_{i \in \text{task } j} \pi_{ji}\, \boldsymbol{\mu}_i^{\text{skill}}, \quad \boldsymbol{\sigma}_j^{\text{task}} = \sqrt{\sum_{i \in \text{task } j} \pi_{ji} \left[(\boldsymbol{\sigma}_i^{\text{skill}})^{\odot 2} + (\boldsymbol{\mu}_i^{\text{skill}} - \boldsymbol{\mu}_j^{\text{task}})^{\odot 2} \right]} \tag{2}$$

Here, π_{ji} is the mixture weight of skill i in task j (details in Appendix B.1), $\odot 2$ denotes element-wise squaring. The task-level measure G_j shares atoms with the global measure G but uses task-specific weights, enabling skill sharing across tasks and forming a hierarchical task–skill knowledge space.

3.3 Knowledge Space Learning via Self-Evolution

We present a unified learning framework where task-centric representations and Dirichlet-Process-based knowledge space co-evolve during continual imitation learning, as summarized in Algorithm 1, inspired by LEGION [12]. Following Section 3.2, we consider the dependency chain from observations $(\boldsymbol{l}, \boldsymbol{o})$ through latent representations z to the knowledge space Θ. To finally get stable knowledge features, a hierarchical variational inference is performed: task-centric latent representations are inferred via VAE [38] for new task learning, and task knowledge is inferred from them through Dirichlet-Process-based models to manage old and new task information. This process gives rise to two variants, T-Stellar and TS-Stellar, corresponding to task-centric and task-skill hierarchical knowledge spaces.

Algorithm 1 Joint Evolution of Task-centric Latents and Knowledge Space

1: Initialize knowledge-space distribution Θ using Dirichlet-Process model
2: Initialize VAE encoder and decoder parameters ϕ, ψ
3: Initialize knowledge buffer $\mathcal{B}_{know}$
4: **for** data $(\boldsymbol{l}_j, \boldsymbol{o}_j, \boldsymbol{a}_j)$ at iteration t **do**
5: $z_j \sim q_\phi(z_j \mid \boldsymbol{l}_j, \boldsymbol{o}_j)$
6: Compute $L_{\text{recon}}(q_\psi(z_j), \boldsymbol{l}_j, \boldsymbol{o}_j), L_{\text{KL}}(z_j, \Theta)$
7: Update VAE parameters $\phi, \psi \leftarrow \phi, \psi - \eta \nabla_{\phi,\psi}\big(L_{\text{recon}} + L_{\text{kl}}\big)$
8: Update knowledge buffer $\mathcal{B}_{know} \leftarrow \mathcal{B}_{know} \cup \{(\boldsymbol{l}_j, \boldsymbol{o}_j)\}$
9: **if** $t \bmod N_{\text{dp}} = 0$ and $t < N_{\text{max}}$ **then**
10: Sample K_{know} latent points $\{z_i\}_{i=1}^{K_{\text{know}}} \sim q_\phi(z \mid (\boldsymbol{l}_j, \boldsymbol{o}_j)),\ (\boldsymbol{l}_j, \boldsymbol{o}_j) \sim \mathcal{B}_{know}$
11: Update knowledge parameters Θ using $\{z_i\}_{i=1}^{K_{\text{know}}}$
12: **end if**
13: **end for**

Task-Centric Representation Learning. We learn task-centric representations z_j from vision-language abstractions using a VAE trained on vision-language data (lines 5-7 in Algorithm 1). The reconstruction loss L_{recon} ensures semantic consistency, while a Dirichlet-Process-aware KL term regularizes the latent space to align with the current knowledge clusters.

For T-Stellar, the KL term is computed as $L_{\text{KL}_j} = \sum_{k=1}^{K_c} p_{jk} L_{\text{KL}_{jk}}$, where p_{jk} denotes the probability of assigning z_j to cluster k, and K_c is the number of task knowledge clusters.

For TS-Stellar, we extend the representation to a hierarchical VAE where task- and skill-level latents are jointly modeled. The reconstruction loss becomes:

$$L_{\text{recon}}^{\text{TS}} = L_{\text{recon}}(z_j^{task}, \boldsymbol{l}_j) + L_{\text{recon}}(z_j^{skill}, \boldsymbol{o}_j), \tag{3}$$

where task latents z_j^{task} decode language goals $\boldsymbol{l}$ and skill latents z_j^{skill} decode visual observations $\boldsymbol{o}$. The corresponding KL regularization follows the HDP structure in Eq. 2:

$$L_{\text{KL}_j}^{\text{TS}} = L_{\text{KL}_j}(z_j^{task}, \theta^{task}) + L_{\text{KL}_j}(z_j^{skill}, \theta^{skill}) \tag{4}$$

Knowledge Space Distribution Learning. To obtain knowledge distribution Θ (lines 8-11 in Algorithm 1), latent representations $\{z_j\}$ are treated as observations of G, and variational inference is performed to approximate the posterior $p(\Theta, G \mid \{z_j\})$. Memoized variational Bayes (memoVB) [39] is adopted for efficient incremental updates, enabling global statistic sharing across tasks.

The task-centric representation learning provides latent features to update the knowledge distribution, while the knowledge space, via KL loss L_{kl}, constrains z_j to infer under task or skill hypotheses. This design establishes a self-evolving cycle that autonomously retains and discovers task knowledge.

3.4 Action-Knowledge-guided Expert Routing

After learning the knowledge space and task-centric latents, we introduce a knowledge-routed MoE module to allocate task-specific action prediction parameters, minimizing interference between unrelated tasks while enhancing sharing among related ones. A diffusion-based MoE action head is adopted referring to MoDE [14]:

$$\text{MoE}(\mathbf{e}, \mathbf{f}_{ro}) = \sum_{i=1}^{N_e} \mathbf{Router}_i(\mathbf{f}_{ro}) \cdot \mathbf{Expert}_i(\mathbf{e}) \tag{5}$$

Here, $\mathbf{e}$ is the input token feature tensor and $\mathbf{f}_{ro}$ the routing features, defined in MoDE as the noise embedding $\mathbf{f}_{ro} = \mathbf{e}_{noise} = \phi(\sigma_\mathbf{t})$, where $\sigma_\mathbf{t}$ is the denoising level. Unlike MoDE's noise-level routing, we guide expert allocation via the knowledge space, enabling continual task learning with task-specific parameter sharing and differentiation. Since DPMM and HDP produce variable numbers of clusters, directly feeding their task distributions into a fixed-dimension network is impractical. Thus, we design two knowledge embedding formats, as shown in Figure 3.

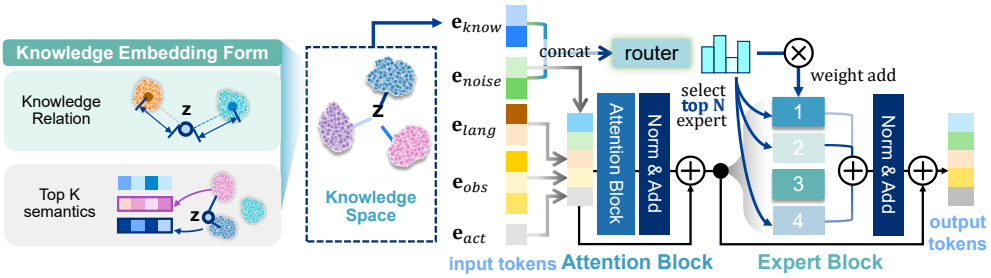


Figure 3: **Knowledge-prior-routed MoE action head.** Two knowledge embeddings, relation and top-K semantic, are computed for expert routing, alongside language, noise, observation and noise action tokens fed into the denoising transformer.

Knowledge Relation Embedding measures the distance between latent z and cluster centers μ_k. The membership probability p_k (details in Appendix B.1), obtained from the knowledge distribution, denotes the posterior assignment of z to cluster k, and is used to compute the relation embedding as:

$$\mathbf{f}_{\text{R}} = \sum_{k=1}^{K_c} p_k \, |z - \mu_k| \tag{6}$$

Top-K Semantic Embedding represents the discrete skill semantic affiliation of z by aggregating learnable embeddings of the top-K assigned clusters:

$$\mathbf{f}_{\text{S}} = \sum_{k \in \mathcal{I}_T} p_k \cdot \text{Embed}(k), \mathcal{I}_T = \text{TopKIndice}(p_1, \ldots, p_K, z) \tag{7}$$

Finally, the embedded knowledge of current task and skill could be represented as:

$$\mathbf{f}_{\text{know}} = [\, z \,\|\mathbf{f}_{\text{R}} \,\|\, \mathbf{f}_{\text{S}} \,] \tag{8}$$

The knowledge embedding $\mathbf{f}_{\text{know}}$ is introduced as an additional guidance token $\mathbf{e}_{know}$, which, together with $\mathbf{e}_{noise}$, are concatenated as input to the router for task-aware expert allocation.

By incorporating knowledge space guidance into expert routing, the model could achieve effective CIL through balanced parameter sharing and isolation, enabling related tasks to benefit from shared experience while preventing catastrophic forgetting across dissimilar ones.

4 Experiments

Our experiments address four questions: (I) How do T-Stellar and TS-Stellar compare with existing VLA and CIL methods? (II) What advantages does the hierarchical task-skill knowledge space in TS-Stellar provide over task-centric modeling in T-Stellar? (III) How do task/skill modeling and knowledge-guided expert routing jointly contribute to continual learning performance? (IV) How does the task knowledge space evolve to discover and preserve skills over time?

4.1 Simulation Evaluation

We evaluate our models on the LIBERO benchmark [40], including LIBERO-goal, LIBERO-long, and LIBERO-30*, a 30-task subset of LIBERO-90 (details in Appendix D.1), covering diverse goals, long-horizon reasoning, and many tasks settings. LIBERO is designed for single-arm manipulation, and demonstrations include static and wrist camera images, with actions defined as end-effector poses and gripper states. We compare against both advanced VLA models and robotic CIL methods.

Baselines. For VLA baselines, we select strong multitask performers on LIBERO and retrain them under the CIL setting for fair comparison. The baselines include: 1) MoDE [14], a diffusion-based Mixture-of-Experts model; 2) UniVLA [1], which learns task-centric representations via latent action pretraining; and 3) π_0 [6] and $\pi_{0.5}$ [2], which leverage large-scale robotic and heterogeneous data for generalization. All methods are evaluated under both from-scratch and cross-embodiment-data pretrained settings (details in Appendix D.1), except $\pi_{0.5}$, which is only evaluated in the pretrained

Table 1: **Overall performance on the LIBERO benchmark.** Baselines are grouped into pretrained and scratch categories. Results are reported in percentages. The highest performance is highlighted in **bold** and the second highest is underline.

tasks	metrics	Scratch VLA Models					Pretrained VLA Models (on cross-embodiment data)					
		MoDE [14]	UniVLA [1]	π_0 [6]	**T-Stellar**	**TS-Stellar**	MoDE [14]	UniVLA [1]	π_0 [6]	$\pi_{0.5}$ [2]	**T-Stellar**	**TS-Stellar**
LIBERO-goal	FWT (↑)	71.0±2.1	3.1±0.7	**82.9±3.9**	81.4±0.2	80.2±2.3	69.7±1.8	**87.5±0.1**	74.9±2.7	84.5±2.5	76.6±3.4	73.6±4.1
	NBT (↓)	36.2±2.8	**3.1±0.7**	49.0±1.0	20.7±2.8	22.2±3.3	23.2±2.9	65.7±6.1	25.6±1.0	38.4±3.7	**19.1±3.6**	28.0±3.7
	AUC (↑)	39.7±1.0	0.4±0.2	45.5±3.1	**61.7±2.2**	60.7±1.2	46.8±2.9	35.2±4.7	55.6±3.3	55.4±4.9	**57.5 ±1.1**	48.1±2.2
	Final SR (↑)	33.1±2.4	0.0±0.0	35.7±0.5	**67.9±0.5**	64.2±1.5	41.6±4.8	20.5±4.7	41.1±1.1	43.8±3.8	**57.0±2.0**	44.1±5.2
LIBERO-long	FWT (↑)	71.6±3.6	6.1±0.1	57.3±3.1	**76.3±4.0**	75.5±0.6	71.1±1.5	70.3±5.6	60.5±2.6	55.5±3.3	80.3±1.6	**81.9±4.1**
	NBT (↓)	46.3±1.6	**6.0±0.1**	48.7±3.7	41.6±4.1	37.3±2.1	46.4±1.1	**31.3±4.0**	34.4±5.5	34.8±1.9	38.0±2.5	37.6±4.4
	AUC (↑)	34.3±4.2	0.9±0.0	22.2±1.0	41.2±0.7	**43.6±1.9**	32.9±1.4	43.6±2.4	35.6±3.7	29.8±5.9	48.6±2.6	**50.8±1.5**
	Final SR (↑)	20.2±2.9	0.0±0.0	12.0±2.2	34.2±2.1	**35.0±3.2**	19.9±1.4	17.1±2.0	24.3±5.1	21.6±5.6	32.2±4.4	**41.7±3.0**
LIBERO-30 *	FWT (↑)	48.2	35.5	72.7	**79.6**	73.3	52.9	**87.9**	70.3	75.0	43.8	84.8
	NBT (↓)	**15.4**	28.1	46.9	31.0	27.8	11.9	57.7	33.5	41.7	**4.1**	37.2
	AUC (↑)	33.6	9.4	31.1	**49.7**	48.3	43.0	36.5	40.8	37.9	38.3	**50.8**
	Final SR (↑)	28.5	12.0	24.9	**42.9**	42.6	40.3	29.6	34.5	35.3	35.8	**41.0**

tasks	metrics	Scratch CIL Methods						Pretrained CIL Methods (on LIBERO-90)					
		ER [8]	SeqLoRA	LoTUS [3]	IsCiL [5]	**T-Stellar**	**TS-Stellar**	ER [8]	SeqLoRA	LoTUS [3]	IsCiL [5]	**T-Stellar**	**TS-Stellar**
LIBERO-goal	FWT (↑)	70.8±1.5	26.2±4.6	44.8±0.3	24.3±5.8	**81.4±0.2**	80.2±2.3	80.3±2.3	80.3±0.6	46.8±4.3	66.8±1.3	80.4±0.3	**82.7±1.6**
	NBT (↓)	33.0±2.0	23.9±4.0	31.3±6.3	**3.0±1.2**	20.7±2.8	22.2±3.3	21.1±0.6	78.8±1.5	37.7±3.9	**7.8±2.5**	12.6±1.0	19.4±2.2
	AUC (↑)	41.8±0.4	8.9±1.1	22.2±5.0	22.0±2.8	**61.7±2.2**	60.7±1.2	60.5±1.8	26.2±1.0	18.5±1.5	60.9±2.8	**66.7±0.5**	64.8±1.6
	Final SR (↑)	34.2±0.9	2.4±0.8	16.3±4.8	18.5±0.5	**67.9±0.5**	64.2±1.5	55.3±1.0	9.5±0.1	10.8±1.1	48.5±2.0	**62.1±1.5**	57.3±5.1
LIBERO-long	FWT (↑)	70.8±6.2	9.6±6.1	11.5±3.0	26.0±6.5	**76.3±4.0**	75.5±0.6	80.3±0.3	62.0±1.3	32.3±2.3	65.8±0.3	79.3±2.1	**81.5±1.3**
	NBT (↓)	47.3±5.5	8.8±5.4	**0.8±1.7**	15.0±4.7	41.6±4.1	37.3±2.1	48.4±2.0	56.0±1.5	**2.8±2.7**	31.5±0.3	36.8±2.8	37.7±2.0
	AUC (↑)	31.0±3.0	2.3±1.4	11.3±4.2	15.2±3.8	41.2±0.7	**43.6±1.9**	40.9±1.1	16.6±0.5	29.4±0.5	41.4±0.1	48.3±3.9	**50.5±1.3**
	Final SR (↑)	18.9±1.1	0.8±0.6	10.5±5.5	6.8±0.8	34.2±2.1	**35.0±3.2**	16.1±0.7	6.0±0.5	31.8±2.3	17.3±0.3	36.3±3.8	**40.9±2.3**

setting, as its main contribution falls on web-scale data pretraining. All models are trained sequentially following the LIBERO task order with Experience Replay (1% memory). For CIL baselines, we include representative replay-, adapter-, and skill-expansion-based methods: 1) Experience Replay (ER) [8]; 2) Sequential Low-Rank Adaptation (SeqLoRA) [15]; 3) LoTUS [3], with incremental skill expansion; and 4) IsCiL [5], with retrieval-based adaptation. All methods are evaluated under both pretrained and scratch settings. Since LoTUS and IsCiL rely on in-domain pretraining, pretrained comparisons are conducted on LIBERO-90, and LIBERO-30* results are omitted.

Metrics. Four metrics are used: FWT (forward transfer), NBT (negative backward transfer), AUC (area under the success rate curve) [5, 3, 41, 40], and Final SR (final success rate). FWT measures forward learning, NBT captures forgetting, AUC reflects overall stability, and Final SR denotes performance after training all tasks (details in Appendix D.1). Each policy is evaluated over 100 trials with 50 initial states. Results are reported as mean $\pm$ standard deviation over 3 training seeds for LIBERO-goal and -long. Due to the high cost of LIBERO-30*, only single-run results are reported.

Results. Table 1 shows that T-Stellar and TS-Stellar achieve the best AUC and final success rates across all scenarios among both VLA and CIL baselines. Notably, in the scratch setting, our methods obtain over 50% average improvement in AUC and Final SR relative to all baselines [1, 14, 6, 2, 8, 3, 5]. Some methods achieve lower NBT mainly due to very low FWT, reflecting the inherent trade-off between forward transfer and backward transfer in continual learning. Among pretrained VLAs, π_0 [6] ($\sim$ 3B), $\pi_{0.5}$ [2] ($\sim$ 3B), and UniVLA [1] ($\sim$7B) show competitive results, likely due to large-scale parameters enabling near-orthogonal gradient updates across tasks in simpler settings. However, without explicit task-specific modeling, their performance is unstable across tasks, while our method achieves the best results with only $\sim$1B parameters (parameter details are in Appendix D.3). Compared with CIL methods, without parameter growth as in LOTUS [3] or IsCiL [5] and with minimal replay data, we achieve 20% improvement in Final SR over all baselines, demonstrating the effectiveness of self-supervised knowledge evolution with task-guided action allocation. Notably, pretraining on LIBERO-90 yields more stable performance than heterogeneous pretraining due to better alignment in embodiment and task distribution, enabling efficient knowledge transfer with minimal adaptation. In contrast, pretraining on over 1k cross-embodiment tasks requires more parameter updates to fit new scenarios. Nevertheless, Stellar VLAs still achieve state-of-the-art performance under heterogeneous pretraining, demonstrating strong knowledge transfer capability.

In response to Question (I), our methods outperform both from-scratch and pretrained baselines, demonstrating strong knowledge accumulation and transfer ability. TS-Stellar further excels on long-horizon tasks, motivating deeper analysis of its hierarchical modeling in the following experiments.

Table 2: **Performance on real-world experiments.** The highest performance is highlighted in **bold** and the second highest is underline.

tasks	metrics	ER [8]	MoDE [14]	UniVLA [1]	π_0 [6]	$\pi_{0.5}$ [2]	T-Stellar	TS-Stellar
Dual arm real world task	FWT (↑)	97.1	15.7	70.0	**98.6**	95.7	**98.6**	**98.6**
	NBT (↓)	21.9	**3.2**	37.1	34.6	25.8	12.4	7.4
	AUC (↑)	79.9	13.0	43.6	72.6	75.6	89.6	**93.4**
	Final SR (↑)	70.0	10.0	21.4	57.1	72.9	84.3	**90.0**



Figure 4: **Success rate curves of real-world experiments and behavior visualization on "Handover Toy" after training on "Pull Stick from bag".** Stellar VLAs maintain high overall success rates, while UniVLA [1] and the π series [6, 2] exhibit clear forgetting during continual learning. MoDE [14], lacking adaptation to dual-arm robots, fails to perform most tasks.

4.2 Real-World Evaluation on Dual-Arm Manipulation

To further validate performance on a new embodiment with compositional manipulation, we conduct real-world dual-arm manipulation experiments on seven tasks involving both single-arm and bimanual operations: "Pick up Stick", "Handover Stick", "Pick up Bag", "Handover Toy", "Pick Toy Place Plate", "Pick Both Object", and "Pull Stick from Bag" (with details in Appendix D.2). Models are trained with a 5% replay rate for stability and evaluated following Section 4.1, with ten trials per task.

Baselines. We compare against all VLA methods [14, 1, 6, 2] reported in Section 4.1, and include ER [8] as a CIL baseline. As described in LoTUS [3] and IsCiL [5], both methods rely on in-domain pretraining. However, our real-robot data involves dual-arm manipulation with substantially different embodiment and scene configurations from LIBERO. Therefore, they are not included in the comparison. This further highlights the advantage of VLA models in adapting to heterogeneous data.

Results. Table 2 show that Stellar VLAs outperform all baselines in AUC, and Final SR while achieve strong NBT and FWT, exhibiting superior CIL ability on dual-arm real-world tasks. TS-Stellar further surpasses T-Stellar across all metrics, suggesting that task-skill modeling is particularly effective for long-horizon skill sequences and complex action mappings, thus addressing Question (II). Figure 4 also visualize the success rate curves of VLA baselines, as well as the behavior of representative VLA models evaluated on "Handover Toy" after training on "Pull Stick from Bag". Results show that our method more effectively mitigates catastrophic forgetting in dual-arm tasks. In "Handover Toy", which requires bimanual skills, UniVLA [1] and $\pi_{0.5}$ [2] exhibit failures at different subtask stages, whereas our method maintains accurate grasping and coordinated handover behaviors.

4.3 Ablation Studies

In this section, we analyze replay efficiency across different budgets, and ablations on the knowledge-space design and expert routing mechanism of Stellar VLA. Specifically, we compare backbone-matched variants of two key reference methods, MoDE [14] and LEGION [12], to evaluate the effectiveness of hierarchical task–skill modeling and knowledge-guided expert routing.

Replay Budget Analysis. Table 1 demonstrates Stellar VLA's strong performance with only 1% replay. To further evaluate replay efficiency across different budgets, we compare ER, T-Stellar, and TS-Stellar on LIBERO-long under 1%, 5%, and 10% replay rates (Table 3). While increased replay improves the performance of all models, both Stellar variants consistently outperform ER in FWT,

Table 3: **Replay budget analysis.**

Buffer	metrics	ER	T-Stellar	TS-Stellar
10%	FWT (↑)	80.8±2.5	83.4±2.2	**85.9**±1.2
	NBT (↓)	8.4±1.4	**1.2**±0.8	2.4±0.2
	AUC (↑)	76.1±1.7	82.3±1.8	**83.8**±1.2
	Final SR (↑)	70.4±1.1	80.1±1.6	**80.2**±1.4
5%	FWT (↑)	74.6±4.2	81.2±2.1	**84.8**±0.9
	NBT (↓)	**3.2**±1.7	5.5±1.3	9.3±1.7
	AUC (↑)	70.4±1.5	76.4±0.7	**77.3**±1.9
	Final SR (↑)	65.0±1.5	72.1±1.8	**73.2**±2.4
1%	FWT (↑)	70.8±6.2	**76.3**±4.0	75.5±0.6
	NBT (↓)	47.3±5.5	41.6±4.1	**37.3**±2.1
	AUC (↑)	31.0±3.0	41.2±0.7	**43.6**±1.9
	Final SR (↑)	18.9±1.1	34.2±2.1	**35.0**±3.2

Table 4: **Knowledge space ablation.**

tasks	metrics	w/o VAE	w/o KS	T-Stellar	TS-Stellar
LIBERO-goal	FWT (↑)	70.8±1.5	76.2±2.0	**81.4**±0.2	80.2±2.3
	NBT (↓)	33.0±2.0	23.2±0.2	**20.7**±2.8	22.2±3.3
	AUC (↑)	41.8±0.4	54.7±2.0	**61.7**±2.2	60.7±1.2
	Final SR (↑)	34.2±0.9	49.8±4.3	**67.9**±0.5	64.2±1.5
LIBERO-long	FWT (↑)	70.8±6.2	72.3±2.3	**76.3**±4.0	75.5±0.6
	NBT (↓)	47.3±5.5	45.3±1.8	41.6±4.1	**37.3**±2.1
	AUC (↑)	31.0±3.0	34.3±3.6	41.2±0.7	**43.6**±1.9
	Final SR (↑)	18.9±1.1	21.4±1.3	34.2±2.1	**35.0**±3.2
LIBERO-30*	FWT (↑)	45.0	45.1	**79.6**	73.3
	NBT (↓)	12.0	**4.9**	31.0	27.8
	AUC (↑)	32.7	38.9	**49.7**	48.3
	SR (↑)	26.3	35.8	**42.9**	42.6

Table 5: **Comparison with LEGION-style model.** LEGION-style model shares the same backbone as Stellar, but only employs DPMM policy conditioning.

tasks	metrics	LEGION-style	T-Stellar	TS-Stellar
LIBERO-goal	FWT (↑)	75.2±2.0	**81.4**±0.2	80.2±2.3
	NBT (↓)	21.3±3.2	**20.7**±2.8	22.2±3.3
	AUC (↑)	54.6±1.5	**61.7**±2.2	60.7±1.2
	Final SR (↑)	54.4±2.0	**67.9**±0.5	64.2±1.5
LIBERO-long	FWT (↑)	75.4±4.0	**76.3**±4.0	75.5±0.6
	NBT (↓)	44.6±3.0	41.6±4.1	**37.3**±2.1
	AUC (↑)	37.9±0.3	41.2±0.7	**43.6**±1.9
	Final SR (↑)	23.0±2.1	34.2±2.1	**35.0**±3.2

Table 6: **Prior-guided routing ablation.** $\mathbf{f}_R$ is knowledge relation embedding, and $\mathbf{f}_S$ is top-K semantic embedding, as in Sec. 3.4.

T-Stellar	w/o ($\mathbf{f}_R + \mathbf{f}_S$)	w/ $\mathbf{f}_R$	w/ $\mathbf{f}_S$	w/ ($\mathbf{f}_R + \mathbf{f}_S$)
FWT (↑)	75.4±4.0	73.5±1.8	**78.3**±2.3	76.3±4.0
NBT (↓)	44.6±3.0	**41.5**±2.5	49.9±1.3	41.6±4.1
AUC (↑)	37.9±0.3	38.1±2.1	37.6±1.3	**41.2**±0.7
Final SR (↑)	23.0±2.1	24.4±2.8	23.9±1.9	**34.2**±2.1
TS-Stellar	w/o ($\mathbf{f}_R + \mathbf{f}_S$)	w/ $\mathbf{f}_R$	w/ $\mathbf{f}_S$	w/ ($\mathbf{f}_R + \mathbf{f}_S$)
FWT (↑)	**78.6**±3.9	76.2±4.0	73.8±1.4	75.5±0.6
NBT (↓)	44.0±2.1	45.0±2.6	42.7±3.1	**37.3**±2.1
AUC (↑)	41.9±3.1	39.1±1.6	38.1±3.5	**43.6**±1.9
Final SR (↑)	30.8±2.0	27.4±1.1	29.5±3.5	**35.0**±3.2

AUC, and Final SR. Notably, Stellar VLA with 5% replay achieves comparable performance to ER with 10%, demonstrating greater replay efficiency.

Knowledge Space Ablation. we replace the Dirichlet-Process-based knowledge space with a standard Gaussian ($\mu = 0, \sigma = 1$) (w/o KS) and remove latent task-centric representation learning (w/o VAE). The w/o KS variant is the same as the ER method compared in Section 4.1. The w/o VAE variant serves as a standard MoE baseline similar to MoDE [14]. Experiments are conducted on all simulation tasks under the scratch setting, as shown in Table 4. The comparison between w/o VAE and w/o KS shows that task-centric representation learning improves continual learning performance. Our designed knowledge space further provides global task distribution information through L_{KL} supervision, reducing feature distribution overlap across tasks and mitigating interference between new and previously learned tasks, ultimately improves almost all four metrics.

LEGION-style Conditioning Ablation. As discussed in Section 1, LEGION [12] is a key reference for Stellar VLA but targets state-based RL. For direct comparison, we construct a LEGION-style model with task-level DPMM conditioning, using the same vision-language encoder and action head as Stellar VLA but without posterior-derived relation and semantic routing features. As shown in Table 5, both Stellar variants improve AUC and Final SR on LIBERO-goal and LIBERO-long, demonstrating the benefits of knowledge-guided routing and hierarchical task–skill modeling.

Knowledge-Guided Expert Routing Ablation. We further evaluate Stellar VLA with individual routing features (defined in Section 3.4) on LIBERO-long, as shown in Table 6. The T-Stellar (w/o ($\mathbf{f}_R + \mathbf{f}_S$)) is the same as LEGION-style model in Table 5. Both full-routing variants achieve higher AUC and Final SR with complete knowledge guidance. The Knowledge Relation Embedding $\mathbf{f}_R$ captures the relevance between z and existing tasks in the global knowledge space, helping the router decide whether new expert combinations are needed. The Top-K Semantic Embedding $\mathbf{f}_S$ encodes the latent task assignment of z, enabling activation of task-relevant expert combinations.

Ablations answer Question (III): Task/skill modeling structures knowledge to reduce interference and promote skill reuse, while knowledge-guided routing selects task-relevant experts. Replay-budget, knowledge-space, and routing ablations demonstrate their combined effectiveness in CIL.

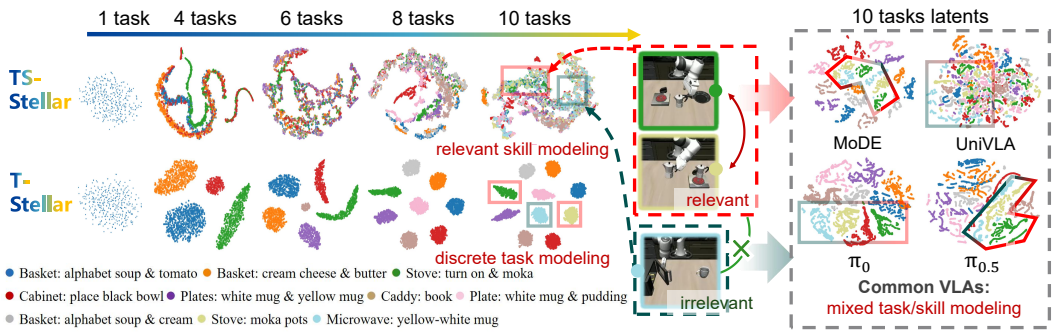


Figure 5: **T-SNE visualization of Stellar VLA** Task latents after 1, 4, 6, 8, and 10 tasks on LIBERO-long are shown. Task names are abbreviated for clarity. T-Stellar models discrete task distributions, and TS-Stellar learn relevant skill across tasks, while other VLAs may mix the task modeling.

4.4 Continual Task Knowledge Evolution Analysis

Knowledge Space Visualization. To illustrate continual skill evolution in Stellar VLA, we visualize VLA latents on LIBERO-long. Figure 5 shows t-SNE plots of latent representations after training on selected tasks. T-Stellar forms distinct task clusters, while TS-Stellar links tasks via shared subskills. For instance, the deep and light green clusters overlap due to the shared subgoal "put moka on stove", while the light blue cluster is separated due to the absence of relevant skills. In contrast, other VLAs [14, 1, 6, 2] either mix these latents or fail to capture task relationship in the feature space.

Subskill Similarity Analysis. To assess whether TS-Stellar's HDP components capture reusable skills, Tasks 3 ("turn on the stove and put the moka pot on it") and 9 ("put both moka pots on the stove") from LIBERO-long are selected, with semantic subskill boundaries manually annotated. Task 3-Sub1/Sub2 denote stove activation and first-pot placement, while Task 9-Sub2/Sub3 denote first- and second-pot placement. Subskill similarity is measured by the Total Variation (TV) distance between inferred component-assignment distributions (Table 7), where lower values indicate higher similarity. In TS-Stellar, the shared first-pot placement shows the highest similarity, followed by second-pot placement due to its different target location, while stove activation is least similar to either placement subskill. In contrast, T-Stellar treats cross-task subskills independently due to its task-level modeling. Consistent with Figure 5, these results demonstrate TS-Stellar's ability to discover reusable subskills without explicit trajectory segmentation supervision. Additional component visualizations and analyses are provided in Appendix E.3.

Table 7: **Cross-task subskill similarity.**

TS-Stellar subtask similarity

TV distance ↓	Task 3-Sub1	Task 3-Sub2
Task 9-Sub2	0.357	0.132
Task 9-Sub3	0.498	0.319

T-Stellar subtask similarity

TV distance ↓	Task 3-Sub1	Task 3-Sub2
Task 9-Sub2	1.000	1.000
Task 9-Sub3	1.000	1.000

Together, these analyses answer Question (IV): T-Stellar develops task-specific components for new tasks, while TS-Stellar discovers shared task–skill structure across related tasks. The resulting reusable components support continual skill acquisition while preserving prior task knowledge.

5 Conclusion

We introduce Stellar VLA, a knowledge space driven policy for continual imitation learning, with two variants: T-Stellar, which constructs task-centric knowledge spaces, and TS-Stellar, which models reusable skills through hierarchical task–skill relations. By using knowledge-posterior features to guide expert selection, our models reduce interference between tasks while enabling continual skill acquisition. Experiments on LIBERO and real-world tasks demonstrate consistent performance gains over strong baselines. Stellar VLA provides useful insights into task knowledge understanding, transfer, and utilization in VLA models. Future work could explore larger skill spaces with dynamic hyperparameter adaptation to scale continual learning for larger VLA models.

Acknowledgments

We thank the reviewers for their valuable comments. This work was supported by National Key R&D Program of China (Grant No.2025YFB4711600). It was also supported in part by the Natural Science Foundation of China under Grant 62225309, U24A20278 and 62361166632, the Explorers Program of Shanghai (Basic Research Funding) under Grant 25TS1415100, in part by the Shanghai Municipal Special Program for Basic Research on General AI Foundation Models under Grant 2025SHZDZX025G01. The authors declare no competing interests.

References

[1] Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Learning to act anywhere with task-centric latent actions. In *Robotics: Science and Systems (RSS)*, 2025.

[2] Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Robert Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, brian ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. In Joseph Lim, Shuran Song, and Hae-Won Park, editors, *Proceedings of The 9th Conference on Robot Learning*, volume 305 of *Proceedings of Machine Learning Research*, pages 17–40. PMLR, 27–30 Sep 2025.

[3] 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)*, pages 537–544. IEEE, 2024.

[4] 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. *arXiv preprint arXiv:2310.05905*, 2023.

[5] 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.

[6] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

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

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

[9] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[10] Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. Quest: Self-supervised skill abstractions for learning continuous control. *Advances in Neural Information Processing Systems*, 37:4062–4089, 2024.

[11] Yating Wang, Haoyi Zhu, Mingyu Liu, Jiange Yang, Hao-Shu Fang, and Tong He. Vq-vla: Improving vision-language-action models via scaling vector-quantized action tokenizers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11089–11099, 2025.

[12] Yuan Meng, Zhenshan Bing, Xiangtong Yao, Kejia Chen, Kai Huang, Yang Gao, Fuchun Sun, and Alois Knoll. Preserving and combining knowledge in robotic lifelong reinforcement learning. *Nature Machine Intelligence*, pages 1–14, 2025.

[13] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[14] Moritz Reuss, Jyothish Pari, Pulkit Agrawal, and Rudolf Lioutikov. Efficient diffusion transformer policies with mixture of expert denoisers for multitask learning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu, editors, *International Conference on Learning Representations*, volume 2025, pages 17247–17275, 2025.

[15] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *In International Conference on Learning Representations*, 2022.

[16] 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*, pages 8748–8763. PMLR, 2021.

[17] Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[18] Brianna Zitkovich, Tianhe Yu, Sichun Xu, Peng Xu, Ted Xiao, Fei Xia, Jialin Wu, Paul Wohlhart, Stefan Welker, Ayzaan Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[19] 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.

[20] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[21] Daehee Lee, Dongsu Lee, TaeYoon Kwack, Wonje Choi, and Honguk Woo. Policy compatible skill incremental learning via lazy learning interface. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen, editors, *Advances in Neural Information Processing Systems*, volume 38, pages 129000–129039. Curran Associates, Inc., 2025.

[22] Likui Zhang, Tao Tang, Zhihao Zhan, Xiuwei Chen, Zisheng Chen, Jianhua Han, Jiangtong Zhu, Pei Xu, Hang Xu, Hefeng Wu, et al. Atomicvla: Unlocking the potential of atomic skill learning in robots. *arXiv preprint arXiv:2603.07648*, 2026.

[23] Yixiao Wang, Yifei Zhang, Mingxiao Huo, Thomas Tian, Xiang Zhang, Yichen Xie, Chenfeng Xu, Pengliang Ji, Wei Zhan, Mingyu Ding, et al. Sparse diffusion policy: A sparse, reusable, and flexible policy for robot learning. In *Conference on Robot Learning*, pages 649–665. PMLR, 2025.

[24] Yuheng Lei, Sitong Mao, Shunbo Zhou, Hongyuan Zhang, Xuelong Li, and Ping Luo. Dynamic mixture of progressive parameter-efficient expert library for lifelong robot learning. *arXiv preprint arXiv:2506.05985*, 2025.

[25] Ralf Römer, Yi Zhang, and Angela P Schoellig. Clare: Continual learning for vision-language-action models via autonomous adapter routing and expansion. *arXiv preprint arXiv:2601.09512*, 2026.

[26] Fanqi Yu, Matteo Tiezzi, Tommaso Apicella, Cigdem Beyan, and Vittorio Murino. Lifelong imitation learning with multimodal latent replay and incremental adjustment. *arXiv preprint arXiv:2603.10929*, 2026.

[27] Zhengyao Jiang, Tianjun Zhang, Michael Janner, Yueying Li, Tim Rocktäschel, Edward Grefenstette, and Yuandong Tian. Efficient planning in a compact latent action space. *arXiv preprint arXiv:2208.10291*, 2022.

[28] Zihao Wang, Shaofei Cai, Zhancun Mu, Haowei Lin, Ceyao Zhang, Xuejie Liu, Qing Li, Anji Liu, Xiaojian Shawn Ma, and Yitao Liang. Omnijarvis: Unified vision-language-action tokenization enables open-world instruction following agents. *Advances in Neural Information Processing Systems*, 37:73278–73308, 2024.

[29] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv preprint arXiv:2312.13139*, 2023.

[30] Wenyao Zhang, Hongsi Liu, Zekun Qi, Yunnan Wang, Xinqiang Yu, Jiazhao Zhang, Runpei Dong, Jiawei He, Fan Lu, He Wang, et al. Dreamvla: a vision-language-action model dreamed with comprehensive world knowledge. *arXiv preprint arXiv:2507.04447*, 2025.

[31] Shengran Hu and Jeff Clune. Thought cloning: Learning to think while acting by imitating human thinking. *Advances in Neural Information Processing Systems*, 36:44451–44469, 2023.

[32] Suneel Belkhale, Tianli Ding, Ted Xiao, Pierre Sermanet, Quon Vuong, Jonathan Tompson, Yevgen Chebotar, Debidatta Dwibedi, and Dorsa Sadigh. Rt-h: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*, 2024.

[33] Tianchen Deng, Yue Pan, Shenghai Yuan, Dong Li, Chen Wang, Mingrui Li, Long Chen, Lihua Xie, Danwei Wang, Jingchuan Wang, Javier Civera, Hesheng Wang, and Weidong Chen. What is the best 3d scene representation for robotics? from geometric to foundation models. *arXiv preprint arXiv:2512.03422*, 2025.

[34] Chongkai Gao, Zixuan Liu, Zhenghao Chi, Junshan Huang, Xin Fei, Yiwen Hou, Yuxuan Zhang, Yudi Lin, Zhirui Fang, Zeyu Jiang, et al. Vla-os: Structuring and dissecting planning representations and paradigms in vision-language-action models. *arXiv preprint arXiv:2506.17561*, 2025.

[35] 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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13890–13900, 2025.

[36] Yuanqi Yao, Siao Liu, Haoming Song, Delin Qu, Qizhi Chen, Yan Ding, Bin Zhao, Zhigang Wang, Xuelong Li, and Dong Wang. Think small, act big: Primitive prompt learning for lifelong robot manipulation. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 22573–22583, 2025.

[37] Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical dirichlet processes. *Journal of the american statistical association*, 101(476):1566–1581, 2006.

[38] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[39] Michael C Hughes and Erik Sudderth. Memoized online variational inference for dirichlet process mixture models. *Advances in neural information processing systems*, 26, 2013.

[40] 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.

[41] Natalia Díaz-Rodríguez, Vincenzo Lomonaco, David Filliat, and Davide Maltoni. Don't forget, there is more than forgetting: new metrics for continual learning. *arXiv preprint arXiv:1810.13166*, 2018.

[42] Jayaram Sethuraman. A constructive definition of dirichlet priors. *Statistica sinica*, pages 639–650, 1994.

[43] Zhenshan Bing, David Lerch, Kai Huang, and Alois Knoll. Meta-reinforcement learning in non-stationary and dynamic environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3476–3491, 2022.

[44] David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

[45] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

[46] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[47] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[48] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models: Open x-embodiment collaboration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6892–6903. IEEE, 2024.

[49] Jun Bai, Minghao Tong, Yang Liu, Zixia Jia, and Zilong Zheng. Understanding and leveraging the expert specialization of context faithfulness in mixture-of-experts llms. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 21938–21953, 2025.

Appendix

A Overview

In the Appendix, we will provide the following details of the method and more results in experiments:

- Sampling and learning process of Dirichlet Process-based models for knowledge evolving in Section B.
- Implementation details of model structure, training process and loss definition in Section C.
- Detailed evaluation settings and engineering analysis in Section D.
- Additional experiments and visualizations in Section E.
- Additional discussions in Section F.

B Dirichlet Process-based Models Details

B.1 Gaussian Mixture Modeling with Dirichlet Processes

Dirichlet Process Sampling. To sample random probability measure G from Dirichlet Process, let G_0 be a base distribution, and $\alpha > 0$ be a concentration parameter. Then Dirichlet Process sampling could be represented as:

$$G \sim DP(\alpha, G_0) \tag{9}$$

A method known as the Stick-Breaking Process (SBP) [42] is employed to generate Dirichlet Process samples. More specifically, the construction introduces an infinite sequence of latent variables that recursively allocate portions of a "stick." First, draw an infinite sequence of independent weights

$$v_k \sim \mathrm{Beta}(1, \alpha), \quad k = 1, 2, \ldots, \tag{10}$$

Where $\mathrm{Beta}(1, \alpha)$ denotes the Beta distribution with shape parameters. The mixing proportions $\{\pi_k\}$ are defined by:

$$\pi_1 = v_1, \qquad \pi_k = v_k \prod_{j=1}^{k-1} (1 - v_j), \quad k \geq 2. \tag{11}$$

Intuitively, v_1 determines the fraction of the stick broken off for the first atom; then v_2 determines the fraction of the remaining stick assigned to the second atom, and so on. In parallel, draw an independent sequence of atoms:

$$\theta_k^* \sim G_0, \tag{12}$$

which represent the locations at which the Dirichlet Process places point masses. Having $\delta_{\theta_k^*}(\theta) \begin{cases} 1, & \text{if } \theta = \theta_k^*, \\ 0, & \text{otherwise.} \end{cases}$, we could finally represent $G \sim DP(\alpha, G_0)$ as:

$$G = \sum_{k=1}^{\infty} \pi_k \, \delta_{\theta_k^*} \tag{13}$$

This formulation guarantees two fundamental properties of the Dirichlet Process: 1) G can contain a potentially infinite number of mixture components, since the stick-breaking process generates an unbounded sequence of weights. 2) the resulting random measure is discrete, since its mass is fully allocated across a countable sequence of atoms.

Gaussian Mixture Modeling and Posterior Inference. In our knowledge space modeling, we adopt a Normal–Wishart base distribution as G_0, ensuring that each task or skill component is represented by a multivariate Gaussian. Given a Dirichlet Process Gaussian Mixture Model with cluster parameters $\theta_k = (\mu_k, \sigma_k)$, the membership probability p_k is defined as the posterior responsibility of cluster k:

$$p_k = p(k \mid z) \propto \pi_k \cdot \mathcal{N}(z \mid \mu_k, \sigma_k),$$

where $\mathcal{N}(\cdot)$ is the Gaussian likelihood under cluster k. The probabilities are normalized as:

$$p_k = \frac{\pi_k \mathcal{N}(z \mid \mu_k, \sigma_k)}{\sum_j \pi_j \mathcal{N}(z \mid \mu_j, \sigma_j)}.$$

This formulation also serves as the foundation for the hierarchical extension (HDP) used in our model. We choose a Dirichlet Process instead of finite parametric models (e.g., Gaussian Mixture Models [43]), as such models require a predefined number of components and cannot flexibly adapt to the expanding knowledge space, whereas Dirichlet Process enables an unbounded number of mixture components through nonparametric clustering.

B.2 Variational inference of the Knowledge Space

Our goal is to approximate posterior $p(\mathbf{v}, \Theta, \beta \mid \mathbf{Z})$ over cluster assignments v_n, component parameters θ_k, and stick-breaking weights β, where $\mathbf{Z} = \{z_1, \ldots, z_N\}$ denotes the observed data, and $\Theta = \{\theta_1, \ldots, \theta_K\}$ represents all component-specific parameters. Direct computation of this posterior is infeasible, so we introduce a factorized variational distribution:

$$q(\mathbf{v}, \Theta, \beta) = \prod_{n=1}^{N} q(v_n) \prod_{k=1}^{K} q(\theta_k) \prod_{k=1}^{K} q(\beta_k), \tag{14}$$

The posterior $p(\mathbf{v}, \Theta, \beta \mid \mathbf{Z})$ could then be approximated by minimizing the Kullback–Leibler (KL) divergence $\mathrm{KL}\big(q(\mathbf{v}, \Theta, \beta) \,\|\, p(\mathbf{v}, \Theta, \beta \mid \mathbf{Z})\big)$. According to [44]), the KL divergence can be rewritten as

$$\begin{aligned} \mathrm{KL}(q\|p) = &- \underbrace{\left(\mathbb{E}_q[\log p(\mathbf{Z}, \mathbf{v}, \Theta, \beta)] - \mathbb{E}_q[\log q(\mathbf{v}, \Theta, \beta)]\right)}_{\text{ELBO}} \\ &+ \log p(\mathbf{Z}). \end{aligned} \tag{15}$$

Since $\log p(\mathbf{Z})$ is constant, minimizing KL is equivalent to maximizing the Evidence Lower Bound (ELBO). The ELBO can be expanded as introduced in [39, 44]:

$$\begin{aligned} \mathrm{ELBO}(q) = &\underbrace{\mathbb{E}_q[\log p(\mathbf{Z} \mid \mathbf{v}, \Theta)]}_{\text{expected data likelihood}} + \underbrace{\mathbb{E}_q[\log p(\mathbf{v} \mid \beta)]}_{\text{assignment prior}} \\ &+ \underbrace{\mathbb{E}_q[\log p(\Theta \mid G_0)]}_{\text{component prior}} + \underbrace{\mathbb{E}_q[\log p(\beta)]}_{\text{stick-breaking prior}} \\ &- \underbrace{\mathbb{E}_q[\log q(\mathbf{v})]}_{\text{assignment entropy}} - \underbrace{\mathbb{E}_q[\log q(\Theta)]}_{\text{component entropy}} \\ &- \underbrace{\mathbb{E}_q[\log q(\beta)]}_{\text{weight entropy}}. \end{aligned} \tag{16}$$

We can iteratively update each factor while holding the others fixed, thereby monotonically increasing the ELBO.

Memoized Variational Bayes. Building on the above principle, Memoized Variational Bayes (memoVB) [39] efficiently performs variational inference for DPMM or HDP models. Specifically, memoVB follows the below steps to update parameters given the observation data z:

- **Local updates:** For each data point z_n, the variational assignment distribution $q(v_n)$ is updated based on the current estimates of the component parameters θ and the stick-breaking weights β. These updates maximize the ELBO with respect to $q(v_n)$, corresponding to the expected data likelihood, the assignment prior, and the assignment entropy in Eq. 16.
- **Global updates:** The component parameters $q(\theta_k)$ and the stick-breaking weights $q(\beta_k)$ are updated using the aggregated information from all data points z weighted by their current assignment probabilities. These updates maximize the ELBO with respect to the global factors, corresponding to the expected data likelihood, the component prior, the stick-breaking prior, and the entropies of $q(\theta_k)$ and $q(\beta_k)$ in Eq. 16.
- **Component management:** This step manages the birth of new components and the pruning or merging of low-contribution ones. New components are added when doing so increases the ELBO, while components with minimal contribution can be pruned or merged. To reduce the computational cost of revisiting many data points, memoVB caches statistics of inactive components, updating only the active ones and thus improving efficiency while preserving the model's flexibility.

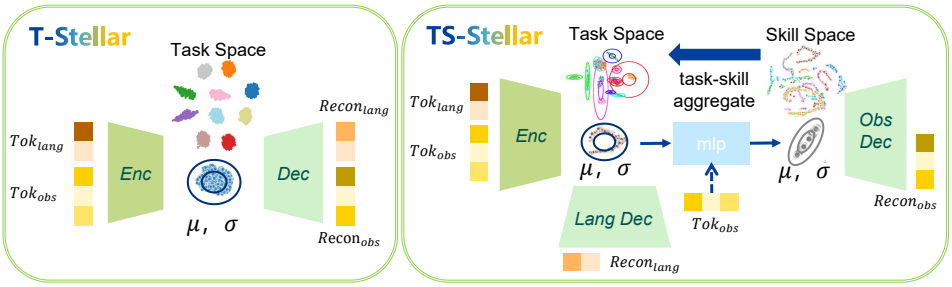


Figure 6: **VAE structure of T-Stellar and TS-Stellar.** T-Stellar adopts a standard VAE framework for task-level modeling, and TS-Stellar introduces a hierarchical structure for joint task–skill modeling.

Through this alternating local-global optimization and dynamic component management, memoVB efficiently maximizes the ELBO while preserving the nonparametric flexibility of the model.

C Implementation Details

C.1 Network Architecture

As discussed in the main paper, Stellar VLA consists of a pretrained vision–language encoder, a VAE encoder–decoder, and a diffusion-based transformer action head with mixture-of-experts (MoE). The network takes language instruction, observation images, and proprioceptive states as input. The language instruction is encoded using CLIP [16] and projected into a 1024-dimensional embedding $\mathbf{e}_{lang}$. Observation images are processed by a FiLM [17]-conditioned ResNet that injects language features, followed by a linear projection producing N image tokens $\mathbf{e}_{obs}$. Proprioceptive inputs are also linearly projected to 1024-dimensional token $\mathbf{e}_{proprio}$.

The VAE encoder is a MLP that takes the concatenation of $\mathbf{e}_{lang}$, $\mathbf{e}_{obs}$, and $\mathbf{e}_{proprio}$ and infers the Gaussian parameters (μ, σ) of the task-centric latent space. z is then sampled via reparameterization. For TS-Stellar, which models both task- and skill-level structure, the latent vector and visual embeddings are further processed by additional linear layers to obtain skill-level Gaussian parameters $(\mu_{\text{skill}}, \sigma_{\text{skill}})$. The architecture of VAE module in T-Stellar and TS-Stellar are shown in Figure 6. For observation reconstruction in the VAE, we use a lightweight MLP decoder and compute the reconstruction loss against the embeddings produced by pretrained vision and language encoders.

The task-centric latent z is 10-dimensional. After combining z with the knowledge-space prior (described in Section 3.4), a linear layer maps the result to the 1024-dimensional knowledge token $\mathbf{e}_{know}$. In the diffusion-based transformer action head, $\mathbf{e}_{know}$, $\mathbf{e}_{lang}$, $\mathbf{e}_{obs}$, $\mathbf{e}_{proprio}$, $\mathbf{e}_{noise}$, and the action noise tokens $\mathbf{e}_{act}$ are jointly fed into the denoising transformer. Final action sequence is acquired from the output $\mathbf{e}_{act}$ through linear layers. Same as MoDE [14], Stellar VLA uses EDM [45]-based diffusion framework for the denoising process. For the transformer MoE, as described in Section 3.4, the tokens $\mathbf{e}_{know}$, $\mathbf{e}_{lang}$, and $\mathbf{e}_{noise}$ are concatenated and used as inputs to the router, which determines expert selection for all tokens. Detailed hyperparameters we used in simulation and real-world experiments are summarized in Table 8.

C.2 Training Losses.

The training objective consists of the VAE losses and the action-head prediction losses. The VAE is optimized using the reconstruction loss L_{recon} and KL divergence L_{kl} as described in Section 3.3.

For the action head, the denoising module is trained via a score-matching loss [46]:

$$L_{\text{SM}} = \mathbb{E}_{\mathbf{a},\,\boldsymbol{\epsilon},\,\sigma}\left[\alpha(\sigma)\,\left\|D_\omega(\mathbf{a}+\sigma\boldsymbol{\epsilon},\ \{Tok_s\},\ \sigma)-\mathbf{a}\right\|_2^2\right], \tag{17}$$

where D_ω is the denoising transformer, $\{Tok_s\}$ denotes all input observation tokens, σ is the noise level, and $\alpha(\sigma)$ is a noise-dependent weighting term.

Table 8: **Network architecture hyperparameters in LIBERO and Real-World Experiments.** The parameters under the Task Knowledge Model part refer to the Dirichlet-Process-based clustering model used in T-Stellar and TS-Stellar, along with their corresponding key hyperparameters.

Hyperparameter	**LIBERO**		**Real-World**	
Proprioceptive States				
Joint State Dim.	7		14	
Gripper State Dim.	2		2	
Vision & Language Encoder				
Image Encoder	FiLM-ResNet50		FiLM-ResNet50	
Static Image Resolution	224*224		240*320	
Wrist Image Resolution	112*112		120*160	
Language Goal Encoder	CLIP ViT-B/32		CLIP ViT-B/32	
Task-Centric VAE	TS-Stellar	T-Stellar	TS-Stellar	T-Stellar
Number of Task Encoder Layers	2	4	2	4
Number of Skill Encoder Layers	2	-	2	-
Hidden Size	1024	1024	1024	1024
Task Latent Size	10	10	10	10
Number of Decoder Layers	3	3	3	3
Task Knowledge Model	TS-Stellar	T-Stellar	TS-Stellar	T-Stellar
Model Type	HDPTopicModel	DPMixtureModel	HDPTopicModel	DPMixtureModel
Concentration (γ_0)	5.0	5.0	5.0	5.0
Covariance Scale (s_F)	1×10^{-5}	1×10^{-5}	1×10^{-5}	1×10^{-5}
Latent KL Weight (β_{kl})	0.001	0.001	0.001	0.001
Birth Threshold (τ_{new})	30	10	30	10
Action Head Transformer				
Number of Layers	12		12	
Number of Attention Heads	8		8	
Embedding Size	1024		1024	
MLP Hidden Size	1024		1024	
Attention Dropout	0.3		0.3	
Residual Dropout	0.1		0.1	
MLP Dropout	0.1		0.1	
Expert Router				
Number of Experts	4		4	
Number of Chosen Top Experts	2		2	
Number of Hidden Layers	1		1	
Linear Action Projection				
Number of Hidden Layers	1		1	
Hidden Size	1024		1024	
Action Chunck Size	10		10	
Action Dim.	7		16	

To prevent expert collapse in the MoE transformer, a load-balancing loss [47] is applied:

$$L_{\text{bal}} = \sum_{i=1}^{N_e} p_i f_i, \tag{18}$$

where N_e is the number of experts, p_i is the router's average assignment probability, and f_i is the empirical usage frequency of expert i. The final training objective can be viewed as a weighted sum of the VAE loss, the action head loss, and the expert loss:

$$L_{\text{total}} = \beta_{\text{recon}}^{\text{lang}} L_{\text{recon}}^{\text{lang}} + \beta_{\text{recon}}^{\text{obs}} L_{\text{recon}}^{\text{obs}} + \beta_{\text{kl}} L_{\text{kl}} + L_{\text{SM}} + \beta_{\text{bal}} L_{\text{bal}}. \tag{19}$$

C.3 Complete Training Process of Stellar VLA.

Section 3.3 in the main text illustrates the co-evolution process of the task-centric representation z and the knowledge space. To more clearly demonstrate how Stellar VLA learns the final manipulation policy during continual robot learning, we present the full training pipeline under a complete data stream of diverse tasks in Algorithm 2.

Algorithm 2 Complete Continual Learning Process of Stellar VLA

Initialize knowledge-space distribution Θ using Dirichlet Process model
Initialize VAE encoder and decoder parameters ϕ, ψ, and action head parameters ω for π_{head}
Initialize data replay buffer $\mathcal{B}$, knowledge buffer $\mathcal{B}_{\text{know}}$
for task j in stream task data $\{\mathcal{T}_j\}_{j=1}^{\infty}$ **do**
    Clear knowledge buffer $\mathcal{B}_{\text{know}} \leftarrow \emptyset$
    for data $(\boldsymbol{l}_j, \boldsymbol{o}_j, s_j^{prop}, \boldsymbol{a}_j)$ in $\mathcal{B} \cup \{\tau_j\}^N$ at iteration t **do**
        $z_j \sim q_\phi(z_j \mid \boldsymbol{l}_j, \boldsymbol{o}_j)$
        Compute $L_{\text{recon}}(q_\psi(z_j), \boldsymbol{l}_j, \boldsymbol{o}_j)$ and $L_{\text{KL}}(z_j, \Theta)$
        $a_j = \pi_{\text{head}}(z_j, \boldsymbol{o}_j, \boldsymbol{l}_j, s_j^{prop}; \omega)$
        Compute action loss L_{SM} and load-balancing loss L_{bal}
        Update VAE parameters $\quad \phi, \psi \leftarrow \phi, \psi - \eta \nabla_{\phi,\psi} L_{\text{total}}$
        Update action head parameters $\quad \omega \leftarrow \omega - \eta \nabla_\omega \big(L_{\text{SM}} + \beta_{\text{bal}} L_{\text{bal}}\big)$
        Update knowledge buffer $\quad \mathcal{B}_{\text{know}} \leftarrow \mathcal{B}_{\text{know}} \cup \{(\boldsymbol{l}_j, \boldsymbol{o}_j)\}$
        if $t \bmod N_{\text{dp}} = 0$ and $t < N_{\text{max}}$ **then**
            Sample K_{know} latent points $\{z_i\}_{i=1}^{K_{\text{know}}} \sim q_\phi(z \mid (\boldsymbol{l}_j, \boldsymbol{o}_j)),\ (\boldsymbol{l}_j, \boldsymbol{o}_j) \sim \mathcal{B}_{know}$
            Update knowledge parameters Θ
        end if
    end for
    Sample small subset $\mathcal{B}_j$ from $\{\tau_j\}^N$
    Update replay buffer $\mathcal{B} \leftarrow \mathcal{B} \cup \mathcal{B}_j$
end for

At the beginning of training, the Dirichlet-Process-based model (DPMM or HDP) initializes the knowledge-space distribution Θ. As training proceeds, the vison and language encoder, task-centric VAE and action head is updated continuously through the final loss L_{total}. Meanwhile, knowledge space maintains its own buffer $\mathcal{B}_{\text{know}}$ that stores data during training. Every N_{dp} iterations, the knowledge space samples K_{know} items from the buffer $\mathcal{B}_{\text{know}}$ and encodes them using the VAE encoder to obtain a set of latent observations $\{z\}$. The Dirichlet-Process-based model then updates its distribution parameters Θ using variational inference described in Section B. Because Θ is updated periodically, the KL term L_{kl} in subsequent VAE updates pulls the latent representation z toward the updated knowledge-space distribution. Through this iterative process, Stellar VLA achieves a mutually reinforcing co-evolution between the task-centric representation and the knowledge space.

Note that the buffer $\mathcal{B}_{\text{know}}$ used for updating knowledge space differs from the memory buffer $\mathcal{B}$ for ER-based CIL described in Section 3.1. It is used solely for updating Dirichlet-Process-based model during training and is cleared after each task, ensuring no additional global storage overhead.

C.4 Training Details.

In continual imitation learning, Stellar VLA is trained on each task with AdamW optimizer for 150 epochs(~10K steps). We use a batch size of 128 with a learning rate of 1e-4 and weight decay of 0.05.

D Evaluation Details

D.1 Simulation Experiments

Correction of Reported Statistics. During final verification, we identified and corrected two statistical reporting errors in Table 1 and the corresponding figures. Specifically, one of the three random-seed runs was inadvertently omitted when aggregating the OXE-pretrained TS-Stellar results on LIBERO-goal and LIBERO-long. Additionally, a transcription error was identified in the standard deviation of the Final SR for LIBERO-90-pretrained TS-Stellar on LIBERO-goal. These corrections do not alter the overall performance trends or the main conclusions of the paper.

LIBERO Benchmark Details. We provide the detailed task order and instructions for the LIBERO experiments reported in the main paper. LIBERO-goal and LIBERO-long are shown in Table 9, and LIBERO-30* is in Table 10. LIBERO-30* uses the first 30 tasks from LIBERO-90, with a small

number of tasks corresponding to the same objectives but different scene configurations. Aside from our work, no prior robotic CL methods have been evaluated on a continual stream of more than 25 tasks in LIBERO-90. Following the official setup, each task is trained using 50 demonstrations.

Table 9: **Task instructions of LIBERO-goal and LIBERO-long benchmark suite.**

Benchmark Suite	Task Order	Task Instructions
LIBERO-Goal	1	open the middle drawer of the cabinet
	2	put the bowl on the stove
	3	put the wine bottle on top of the cabinet
	4	open the top drawer and put the bowl inside
	5	put the bowl on top of the cabinet
	6	push the plate to the front of the stove
	7	put the cream cheese in the bowl
	8	turn on the stove
	9	put the bowl on the plate
	10	put the wine bottle on the rack
LIBERO-Long	1	put both the alphabet soup and the tomato sauce in the basket
	2	put both the cream cheese box and the butter in the basket
	3	turn on the stove and put the moka pot on it
	4	put the black bowl in the bottom drawer of the cabinet and close it
	5	put the white mug on the left plate and put the yellow and white mug on the right plate
	6	pick up the book and place it in the back compartment of the caddy
	7	put the white mug on the plate and put the chocolate pudding to the right of the plate
	8	put both the alphabet soup and the cream cheese box in the basket
	9	put both moka pots on the stove
	10	put the yellow and white mug in the microwave and close it

Table 10: **Task instructions of LIBERO-30* benchmark suite.**

Task Order	Scene	Task Instruction
1	KITCHEN SCENE10	close the top drawer of the cabinet
2	KITCHEN SCENE10	close the top drawer of the cabinet and put the black bowl on top of it
3	KITCHEN SCENE10	put the black bowl in the top drawer of the cabinet
4	KITCHEN SCENE10	put the butter at the back in the top drawer of the cabinet and close it
5	KITCHEN SCENE10	put the butter at the front in the top drawer of the cabinet and close it
6	KITCHEN SCENE10	put the chocolate pudding in the top drawer of the cabinet and close it
7	KITCHEN SCENE1	open the bottom drawer of the cabinet
8	KITCHEN SCENE1	open the top drawer of the cabinet
9	KITCHEN SCENE1	open the top drawer of the cabinet and put the bowl in it
10	KITCHEN SCENE1	put the black bowl on the plate
11	KITCHEN SCENE1	put the black bowl on top of the cabinet
12	KITCHEN SCENE2	open the top drawer of the cabinet
13	KITCHEN SCENE2	put the black bowl at the back on the plate
14	KITCHEN SCENE2	put the black bowl at the front on the plate
15	KITCHEN SCENE2	put the middle black bowl on the plate
16	KITCHEN SCENE2	put the middle black bowl on top of the cabinet
17	KITCHEN SCENE2	stack the black bowl at the front on the black bowl in the middle
18	KITCHEN SCENE2	stack the middle black bowl on the back black bowl
19	KITCHEN SCENE3	put the frying pan on the stove
20	KITCHEN SCENE3	put the moka pot on the stove
21	KITCHEN SCENE3	turn on the stove
22	KITCHEN SCENE3	turn on the stove and put the frying pan on it
23	KITCHEN SCENE4	close the bottom drawer of the cabinet
24	KITCHEN SCENE4	close the bottom drawer of the cabinet and open the top drawer
25	KITCHEN SCENE4	put the black bowl in the bottom drawer of the cabinet
26	KITCHEN SCENE4	put the black bowl on top of the cabinet
27	KITCHEN SCENE4	put the wine bottle in the bottom drawer of the cabinet
28	KITCHEN SCENE4	put the wine bottle on the wine rack
29	KITCHEN SCENE5	close the top drawer of the cabinet
30	KITCHEN SCENE5	put the black bowl in the top drawer of the cabinet

Baseline Details. For VLA baselines, all models are trained with 1% data replay. MoDE [14] follows the same setting as ours with 10K steps per task. For π_0 and $\pi_{0.5}$, we follow the continual learning protocol in [7], using LoRA with AdamW, batch size 8, and 10K steps per task. For UniVLA [1],

Table 11: **Cross-Embodiment Pretraining Data Split for Stellar VLAs.**

	BC-Z	BridgeV2	CMU Play-Fusion	Google Fractal	DOBB-E
Weight	0.2708	0.1979	0.1042	0.1667	0.2604

which does not provide a continual learning setup, we construct a baseline based on its multitask training configuration (30K total steps on 10 tasks), resulting in 3K steps per task with batch size 128. All methods follow standard configurations from prior continual learning and VLA literature, with minor adaptations for our setting. Each model is trained to stable forward convergence under its respective regime, as evidenced by consistent FWT performance (Table 1), ensuring a fair and practically grounded comparison.

For CIL baselines, we include representative continual learning baselines covering replay-, adapter-, and skill-expansion approaches. ER [8] adopts the same network architecture as Stellar VLA but relies solely on data replay without task representation learning or knowledge space modeling, and is therefore also used as the w/o VAE ablation in Sec. 4.3. SeqLoRA [15] also shares the same architecture as Stellar VLA, inserting adapters into all linear layers and merging them into the base model after training each task. For LOTUS [3], although results on LIBERO-goal are reported in the original paper, the reported setting corresponds to pretraining on the first six tasks followed by continual learning on the remaining four tasks. To ensure a consistent pretraining protocol, we conduct pretraining on LIBERO-90 and evaluate full 10-task continual learning on both LIBERO-goal and LIBERO-long. For IsCiL [5], we reproduce its experiments following the official implementation. We further note that latent replay approaches such as MLR [26] are not included in our comparisons, as their reported evaluation protocol differs from our unified full-task continual learning setting.

Cross-Embodiment Pretraining Details. For cross-embodiment pretraining, we follow MoDE [14] to pretrain for 300k steps on a subset of Open X-Embodiment(OXE) dataset [48], as shown in Table 11, with over 1,000 distinct language-conditioned tasks.. For UniVLA, we use the official model pretrained on full OXE dataset for action representations with BridgeV2 post-training. For π_0 [6] and $\pi_{0.5}$ [2], we also use the official pretrained model. π_0 is pretrained on large-scale proprietary robot data, together with a subset of the OXE dataset, while $\pi_{0.5}$ further incorporates large-scale internet multimodal data.

Metrics Formulations. FWT (forward transfer), NBT (negative backward transfer), AUC (area under the success rate curve), and Final SR (final average success rate) are computed from the success rate matrix R, where $R_{i,j}$ is the success on task j after learning i tasks. Formally, FWT $= \frac{1}{M}\sum_{m=1}^{M} R_{m,m}$, NBT $= \frac{1}{M}\sum_{m=1}^{M}\frac{1}{M-m}\sum_{q=m+1}^{M}(R_{m,m} - R_{q,m})$, AUC $= \frac{1}{M}\sum_{m=1}^{M}\frac{1}{M-m+1}\left(R_{m,m} + \sum_{q=m+1}^{M} R_{q,m}\right)$, Final SR $= \frac{1}{M}\sum_{j=1}^{M} R_{M,j}$.

D.2 Real-World Experiments

Experiment Setup. We conduct real-world experiments on an AGIBOT G1 robot equipped with a two-finger gripper, where both arms are controlled via 7-DoF absolute joint positions. Seven tasks are designed, as shown in Figure 7: 1) "Pick up Stick": Pick up the magic wand with right gripper. 2) "Handover Stick": Pick up the magic wand with right gripper and transfer it to the left gripper. 3) "Pick up Bag": Pick up the bag on table with both grippers. 4) "Handover Toy": Pick up the toy with right gripper and transfer it to the left gripper. 5) "Pick Toy Place Plate": Pick up the toy with right gripper and place it on the plate. 6) "Pick Both Object": Pick up the toy and the magic wand at the same time. 7) "Pull Stick from Bag": Extract the magic wand from the bag. Each task is trained with 50 expert demonstrations.

Policy Training and Evaluation. For dual-arm manipulation, both proprioceptive and predicted states are represented as 14-dimensional absolute joint positions (7 per arm) with 2-dimensional gripper states, and visual inputs consist of images from a head camera and two wrist cameras. All methods are pretrained on cross-embodiment data. For Stellar VLA, to adapt to different action dimensions, the final projection layer (linear layer in Figure 2) is replaced accordingly while keeping the backbone unchanged. Since dual-arm manipulation introduces additional eye-in-hand camera, the number of visual tokens fed into the action head is increased. In Stellar VLA, to enhance spatial awareness in real-world settings, each camera contributes 4 image tokens. For these new token

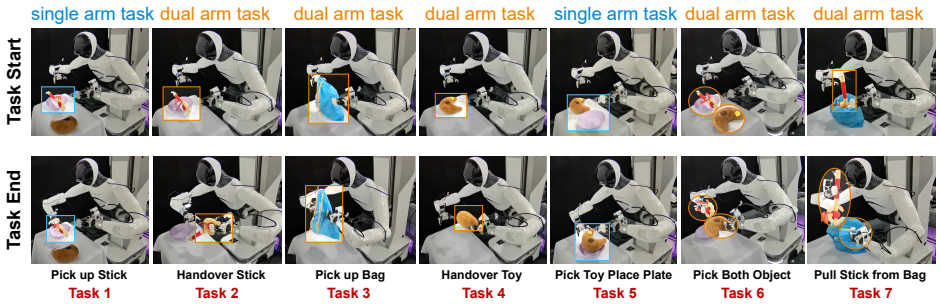


Figure 7: **Real-world Dual-Arm Task Settings.** We design 7 tasks involving both single-arm and bimanual manipulation, covering skills such as grasping, placing, handover, and coordinated extraction, where one arm provides stabilization while the other executes the extraction (Task 7).

Table 12: **Inference Time and Model Parameters Comparison.** Inference time are reported for the average time of predicting each action step on NVIDIA RTX 4090.

Method	Params	Test Time		
		Total	Network Inference	Knowledge token computing
MoDE [14]	0.89B	26.43ms	26.43ms	–
UniVLA [1]	7.55B	143.34ms	143.34ms	–
π_0 [6]	3.29B	83.36ms	83.36ms	–
$\pi_{0.5}$ [2]	3.40B	89.82ms	89.82ms	–
T-Stellar	0.98B	27.20ms	27.10ms	0.10ms
TS-Stellar	0.98B	27.60ms	27.47ms	0.13ms

inputs, the pretrained image encoder and projection layer are reused to obtain token embeddings, and sinusoidal positional encodings are applied based on the spatial positions of the tokens.

During evaluation, we test on 10 random initial states with slight variations in lighting conditions, without requiring strict environmental control.

D.3 Computational and Engineering Analysis

Training and Inference Cost Comparison. To further demonstrate that our method can be trained end-to-end with efficiency comparable to standard policy learning models, we summarize the model size and inference time of Stellar VLAs and VLA baselines in Table 12. Compared to MoDE [14], the introduction of knowledge space modeling incurs negligible overhead in both parameter count and inference latency. Moreover, despite having only about one-third the parameters of the π series [6, 2] and one-seventh of UniVLA [1], our model achieves strong continual learning performance without relying on additional parameter growth, highlighting the strong continual learning capability of Stellar VLA under a parameter-efficient design.

Training time is not directly comparable across methods due to differences in memory requirements. As UniVLA and π_0 both exceed 3B parameters, a single NVIDIA H200 GPU is insufficient for training. Therefore, we report their results using 2 H200 GPUs. Following the official implementations, UniVLA [1] is trained for about 32 hours on 2 H200 GPUs for 10 LIBERO tasks, and π series models [6, 2] take about 10 hours. In contrast, Stellar VLA is trained about 16 hours on a single H200 GPU for 10 LIBERO-goal tasks. As our policy network is adapted from MoDE [14], we further provide a fair single-GPU training time comparison between our method and MoDE in Table 13. Results show that updating the knowledge space introduces only about a 2% increase in training time.

Storage Analysis of Continual Learning with Buffer Replay. Although Stellar VLA does not rely on additional network parameters, it still requires a small amount of data replay during continual learning. Based on the official LIBERO dataset size [40], a 1% replay ratio introduces approximately 10MB of additional storage per task. However, even for 30 tasks, this results in less than 500MB of extra storage, which remains a relatively modest disk requirement.

Table 13: **Training Time Comparison on LIBERO-goal.**

Method	Total Training Time	Network Training	Knowledge Space Update
MoDE [14]	14.65h	14.65h	–
T-Stellar	16.22h	15.97h	0.25h
TS-Stellar	16.24h	15.98h	0.26h

E Additional Experiments

E.1 Multitask Learning Results on LIBERO.

We report multitask results on LIBERO-goal and LIBERO-long to demonstrate that the evaluated VLA models [1, 14, 6, 2] are strong baselines, as shown in Table 14. Since MoDE [14] only provides results on LIBERO-long, we reproduce it on LIBERO-goal for completeness. The results show that Stellar VLAs also achieve strong performance in the multitask setting.

Table 14: **Success rate on the LIBERO benchmark in the multitask setting**. Results are reported as percentages, with the best scores shown in **bold** and second scores shown in underline. All evaluated models perform well, with Stellar VLA variants achieving the best overall performance.

tasks	UniVLA [1]	MoDE [14]	π_0 [6]	$\pi_{0.5}$ [2]	T-Stellar	TS-Stellar
LIBERO-goal	95.6	96.6	95.8	**98.0**	96.5	<u>96.7</u>
LIBERO-long	92.0	92.0	85.2	92.4	**94.4**	<u>94.3</u>

E.2 More Ablations

Ablation on Task Orderings. Task ordering naturally affects continual learning dynamics, as it implicitly defines a learning curriculum over tasks with varying difficulty and structure. Following common practice [7, 3, 5], we adopt the default task ordering provided by the LIBERO benchmark in our main experiments. To further analyze the effect of task ordering, we evaluate LIBERO-90-pretrained TS-Stellar on LIBERO-long under alternative task orderings, as shown in Table 15. Specifically, in addition to the default ordering (0123456789), we consider an inverse ordering (9876543210) and a random ordering (4687312095).

Different orderings lead to variations across metrics, but the overall performance remains stable. The inverse ordering shows slightly lower FWT but reduced NBT, indicating less interference, while the random ordering achieves comparable AUC with a marginal improvement in final success rate. These differences are largely driven by the implicit curriculum induced by task ordering, particularly the distribution of task difficulty and structural similarity. Despite these variations, Stellar VLA maintains consistently strong AUC and final success rate across orderings.

Ablation of Hyperparameters in Dirichlet-Process Model. The impact of probabilistic model parameters in the knowledge space is also investigated, with particular focus on the most influential factor for clustering performance, the covariance scale s_F, as reported in Table 16. The parameter s_F controls the effective resolution of the latent Gaussian components by scaling their covariance, thereby determining the compactness of each cluster and the degree of separation between different clusters in the knowledge space. A larger s_F produces broader components that cover a wider region of the latent space. In contrast, a smaller s_F yields more concentrated components that fit the data more tightly, increasing the tendency to split into multiple clusters. As shown in the results, both overly fragmented and overly unified knowledge spaces lead to degraded performance, with the effect being more pronounced in the task-only modeling (T-Stellar) setting.

Knowledge Space Modeling Ablation. To evaluate the advantages of nonparametric Bayesian models (DPMM/HDP) over fixed-capacity alternatives, Table 17 compares knowledge space modeling with a fixed-K Gaussian mixture model (Fixed-K GMM) against the w/o KS ablation and Stellar variants under the scratch setting. Fixed-K GMM improves AUC and Final SR over w/o KS but underperforms Stellar variants on both LIBERO-goal and LIBERO-long, as its fixed number of components may lead to substantial semantic shifts as new tasks arrive. These results highlight the benefits of adaptively adjusting knowledge components throughout sequential task learning.

Table 15: **Ablations of task ordering for TS-Stellar on LIBERO-long.** Default ordering is 0123456789, inverse ordering is 9876543210, and random ordering is 4687312095.

tasks	metrics	Task Ordering		
		inverse	random	default
LIBERO-long	FWT (↑)	76.8	82.8	81.5
	NBT (↓)	28.7	38.4	37.7
	AUC (↑)	53.2	51.8	50.5
	Final SR (↑)	41.0	43.3	40.9

Table 16: **Ablations of Hyperparameters in Dirichlet-Process Model.** $s_F = 1 \times 10^{-5}$ is the configuration we used in the main experiments. The best performance is highlighted in **bold** for T- and TS- Stellar ablations respectively.

tasks	metrics	T-Stellar			TS-Stellar		
		$s_F = 1 \times 10^{-3}$	$s_F = 1 \times 10^{-10}$	$s_F = 1 \times 10^{-5}$	$s_F = 1 \times 10^{-3}$	$s_F = 1 \times 10^{-10}$	$s_F = 1 \times 10^{-5}$
LIBERO-long	FWT (↑)	68.4±1.2	76.0±3.8	76.3±4.0	**79.0±1.1**	73.6±2.8	75.5±0.6
	NBT (↓)	**37.4±5.3**	47.0±5.7	41.6±4.1	44.3±2.4	38.3±4.9	**37.3±2.1**
	AUC (↑)	36.6±5.4	37.3±1.7	**41.2±0.7**	42.6±0.5	41.5±0.2	**43.6±1.9**
	Final SR (↑)	26.2±2.8	25.7±2.9	**34.2±2.1**	29.8±1.3	33.9±0.3	**35.0±3.2**

E.3 More Visualizations on LIBERO

Success Rates Visualization. We visualize the success rate curves under the pretraining setting on LIBERO-goal and LIBERO-long in Figure 8 and Figure 9, respectively. The success rate variations of VLA models on LIBERO-30* are visualized as heatmaps in Fig. 10. Compared to larger VLA baselines [1, 6, 2], Stellar VLA exhibits early-stage fluctuations but improves as more tasks are accumulated. A typical example is UniVLA [1] in Figure 9 and Figure 10, which maintains high performance across all tasks in the initial stage, benefiting from its large model capacity, but experiences a clear performance drop starting from Task 6. In contrast, the smaller-scale Stellar VLA maintains stable forward transfer and overall success rate after Task 5.

Compared to CIL baselines, Stellar VLA achieves strong average performance. Notably, on LIBERO-goal, both T-Stellar and TS-Stellar show a clear "drop-then-recovery" pattern on Task 2. As shown in Table 9, Task 2 and Task 9 share similar skills, leading to a performance boost on Task 2 during learning of Task 9, reflecting the task discrimination ability of Stellar VLA. Nevertheless, the initial performance degradation in early-stage continual learning indicates a limitation of the current design, which we further analyze via knowledge visualization in the following knowledge evolution section.

More Visualization for Knowledge Evolution. Figure 11 further compares the Gaussian components learned by the Dirichlet-Process-based model with the ground-truth task labels. For TS-Stellar, the learned components effectively form a self-supervised skill-level partition as discussed in Section 4.4. For T-Stellar, the final components largely align with the true task labels, demonstrating accurate modeling of task-level structure.

However, in the early stages of continual learning, the knowledge model may partition a small number of tasks into more sub-clusters (as observed in Task 4 and Task 6 for TS-Stellar and T-Stellar). This tendency can lead the MoE to allocate relatively complex expert combinations to earlier tasks. As new tasks introduce additional expert compositions, this may slightly affect previously learned experts assigned for certain sub-stages, resulting in some performance degradation on earlier tasks. This effect could potentially be alleviated by dynamically adjusting clustering parameters and the weighting of knowledge space learning objectives in future work.

Expert Usage Analysis. To examine whether the proposed knowledge-guided expert routing enables task specialization, we follow the visualization procedure of [49] to analyze expert activation patterns. After continual training on LIBERO-long, we record the expert-selection frequencies of Stellar VLA across all previously learned tasks. Expert activations from each episode are concatenated into a feature vector, and 20 episodes are sampled per task. Results are visualized using t-SNE, as shown in Figure 12. Both variants show task-dependent expert patterns: TS-Stellar has more overlap, while T-Stellar forms mostly distinct clusters. Because two-dimensional projections can

Table 17: **Knowledge space modeling ablation.**

Suite	Metric	w/o KS	Fixed-K GMM	T-Stellar	TS-Stellar
LIBERO-goal	FWT ↑	76.2±2.0	74.4±2.5	**81.4**±0.2	80.2±2.3
	NBT ↓	23.2±0.2	**20.0**±1.7	20.7±2.8	22.2±3.3
	AUC ↑	54.7±2.0	56.9±2.2	**61.7**±2.2	60.7±1.2
	Final SR ↑	49.8±4.3	58.6±2.4	**67.9**±0.5	64.2±1.5
LIBERO-long	FWT ↑	72.3±2.3	75.4±3.2	**76.3**±4.0	75.5±0.6
	NBT ↓	45.3±1.8	46.7±4.3	41.6±4.1	**37.3**±2.1
	AUC ↑	34.3±3.6	37.1±2.0	41.2±0.7	**43.6**±1.9
	Final SR ↑	21.4±1.3	29.3±2.2	34.2±2.1	**35.0**±3.2

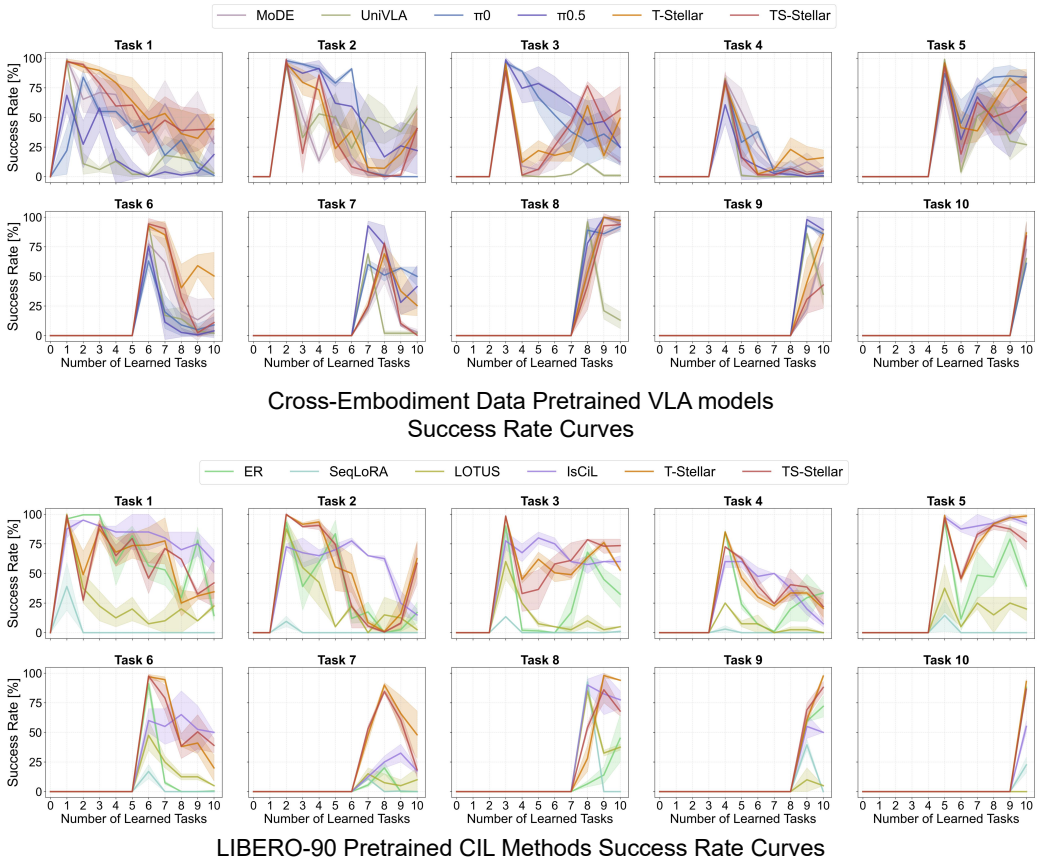
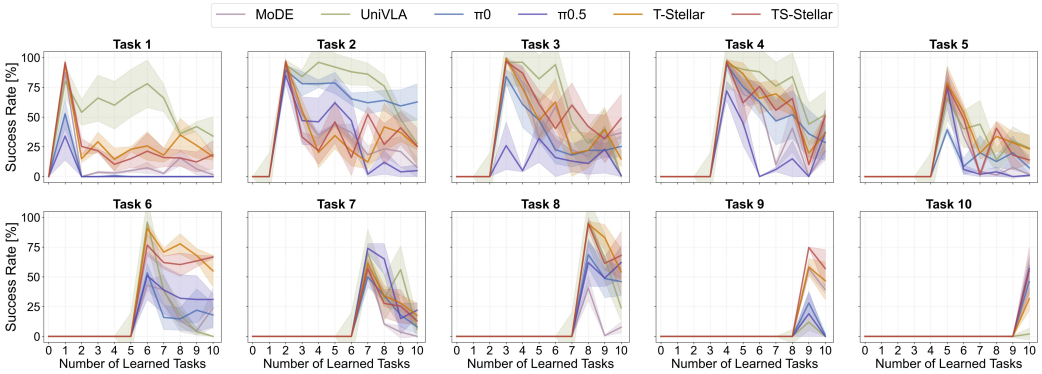


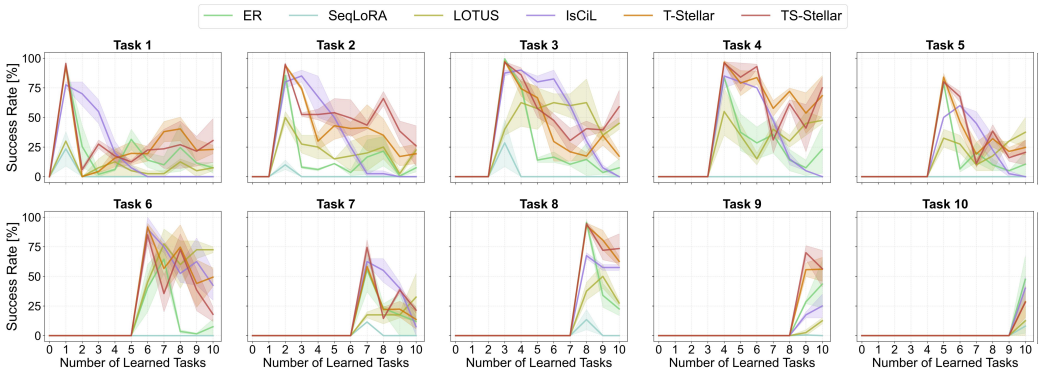
Figure 8: **Success rate curves of all pretrained models on LIBERO-goal.** Shaded regions denote the standard deviation of success rates. Stellar VLAs exhibit more stable overall performance, particularly after learning five tasks in the continual setting.

distort high-dimensional distances, Table 18 compares the routing distributions and their association with forgetting across methods.

Further Quantitative Analysis for Expert Routing. Table 18 compares inference-time expert-routing TV distances and task NBT across three LIBERO-long task pairs with different skill similarities. As defined in Table 9, Tasks 1, 2, and 8 share similar skills, whereas Task 7 is unrelated. The relative separation ratio measures the TV distance of the unrelated pair relative to the average of the two related pairs. Results show that TS-Stellar achieves the highest relative separation ratio, indicating more skill-aware expert allocation and stronger forgetting mitigation. In contrast, T-Stellar exhibits larger expert distribution differences due to independent task modeling, while ER shows less discriminative routing without knowledge modeling. These results suggest that knowledge-guided routing, combined with hierarchical task–skill modeling, better captures task relationships and mitigates forgetting in skill-compositional scenarios.



Cross-Embodiment Data Pretrained VLA models
Success Rate Curves



LIBERO-90 Pretrained CIL Methods Success Rate Curves

Figure 9: **Success rate curves of all pretrained models on LIBERO-long.** Shaded regions denote the standard deviation of success rates. Stellar VLAs exhibit more stable overall performance, particularly after learning four tasks in the continual setting

Table 18: **Expert overlap and forgetting analysis on LIBERO-long.**

	TS-Stellar		T-Stellar		ER	
Task A–B	TV ↓	NBT ↓	TV ↓	NBT ↓	TV ↓	NBT ↓
1–8 (related)	0.027	-15.3	0.067	-16.6	0.016	-20.0
2–8 (related)	0.012	-25.0	0.021	10.6	0.011	-15.2
2–7 (unrelated)	0.063	5.0	0.062	-14.6	0.024	10.3
Relative separation ratio ↑	**3.231**	–	1.409	–	1.780	–
Average NBT ↓	–	**-11.7**	–	-6.9	–	-8.3

E.4 Real-World Performance Analysis.

Behavior Rollouts. To illustrate policy behavior under continual learning, we present head-camera visualizations of TS-Stellar and $\pi_{0.5}$ [2]. The visualizations show rollouts on "Handover Stick" and "Pick Toy Place Plate" after training on the final task "Pull Stick from Bag", as shown in Figures 13 and 14. We observe that $\pi_{0.5}$ exhibits clear failures in bimanual or single-arm manipulation after successfully grasping the object, while TS-Stellar maintains accurate handover and placing behaviors. This suggests that TS-Stellar better preserves prior task knowledge during continual learning. More visualizations across tasks are included in the supplementary zip archive.

Generalization Analysis. To probe the limits of model capability, we conduct a preliminary clutter generalization experiment on "Pick Toy Place Plate". We vary initial object positions under two different scene configurations, with five trials per setup. The results and example rollouts are shown in Figure 15. For most VLA baselines [14, 1, 6], performance on the raw "Pick Toy Place Plate"

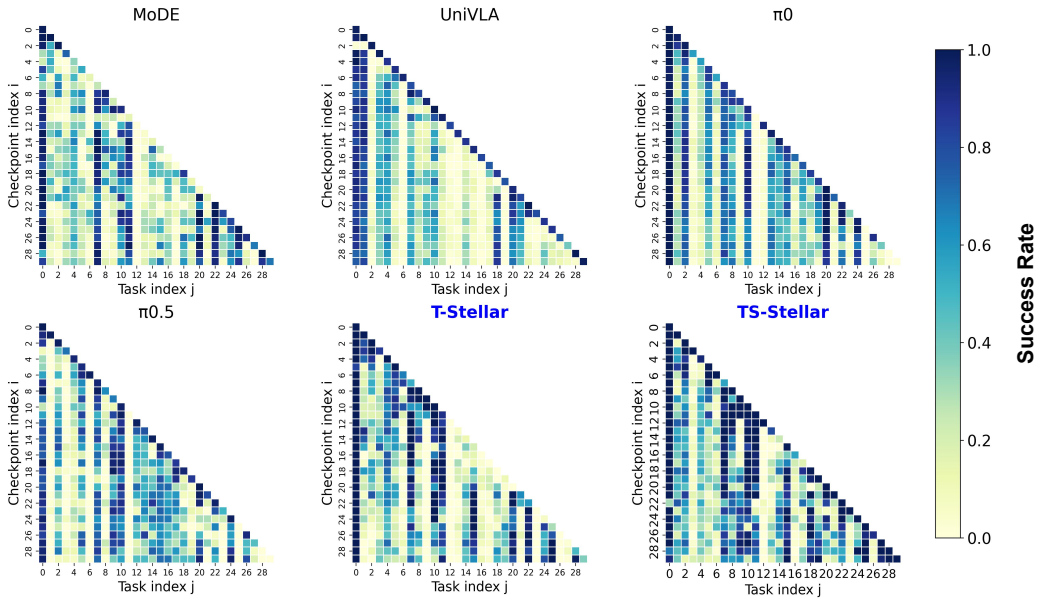


Figure 10: **Success rates comparison of VLA models on LIBERO-30*.** TS-Stellar shows more prominent overall coloring in the heatmap, indicating stronger overall continual learning performance.

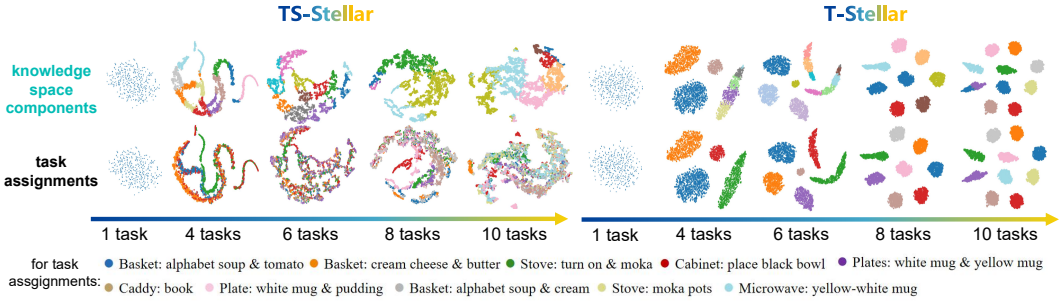


Figure 11: **Visualization of knowledge space components in Stellar VLA.** Task labels are shown in the bottom row, while dynamic knowledge-space components are only color-coded. Comparing knowledge space clusters with task labels, TS-Stellar groups related tasks into shared skill components, and T-Stellar learns components that largely align with true task assignments.

task already degrades after training on all tasks (as indicated in Figure 4), leading to relatively poor generalization. Stellar VLA achieves higher overall success rates in cluttered settings, while $\pi_{0.5}$ also remains competitive, likely benefiting from large-scale vision-language data pretraining.

However, this generalization ability may also partly stem from behavioral fitting to the training data. We observe that success tends to concentrate around object configurations frequently seen during training. Future work could explore mechanism-driven parameter steering guided by the task knowledge space, enabling more robust few-shot or even zero-shot generalization and further leveraging the interpretability of Stellar VLA.

F Discussions

F.1 Discussion on Hierarchical Task–Skill Modeling

As in main-text experiments, TS-Stellar outperforms T-Stellar on LIBERO-long and real-world tasks but can underperform on LIBERO-goal and LIBERO-30*. Figure 11 shows that TS-Stellar can create more knowledge components than the number of tasks early in training. These observations motivate a controlled study of component allocation on OXE-pretrained LIBERO-goal, where hierarchical

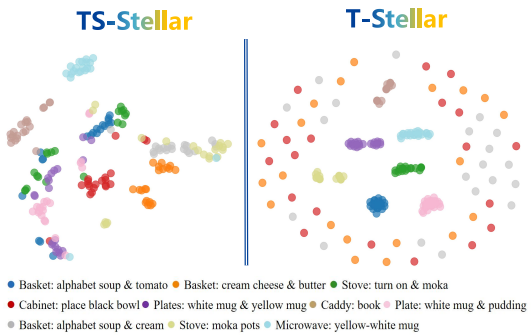


Figure 12: **T-SNE visualization of knowledge-routed expert activation patterns on LIBERO-long.** Both TS-Stellar and T-Stellar exhibit a certain degree of task-specific expert routing patterns.



Figure 13: **Behavior visualization on "Handover Stick" after training on "Pull Stick from Bag".** TS-Stellar accurately completes both the single-arm grasping and bimanual handover sub-stages, whereas $\pi_{0.5}$ [2] fails to precisely align the gripper poses during the bimanual handover stage.

task–skill modeling underperforms task-only modeling.Table 19 shows that limiting component creation improves TS-Stellar across all four metrics on LIBERO-goal. However, it still underperforms T-Stellar, suggesting that over-segmentation is an important but not the only contributing factor. Learning transferable skill representations from visual features may also be challenging, particularly for tasks requiring fine spatial distinctions, whereas skill sharing can be more beneficial for longer, compositional tasks. This is consistent with the stronger performance of TS-Stellar on LIBERO-long.

Table 19: **Effect of controlled component allocation for TS-Stellar on LIBERO-goal.**KS comp indicates the number of knowledge components after learning all tasks.

Method	KS comp	FWT ↑	NBT ↓	AUC ↑	Final SR ↑
T-Stellar	10	76.6±3.4	**19.1**±3.6	**57.5**±1.1	**57.0**±2.0
TS-Stellar	7	73.6±4.1	27.3±3.7	48.1±2.2	44.1±5.2
Controlled-K TS-Stellar	3	**78.1**±1.1	24.8±2.3	53.9±1.9	54.4±2.5

F.2 Limitations and Future Works

VLA Model and Knowledge Space Scaling up. As detailed in Section C, Stellar VLA adopts a small pretrained VLM for vision–language encoding, with an overall model size of around 1B parameters. This limited capacity may hinder the model's ability to interpret subgoal-level information embedded in language or visual observations, thereby restricting its few-shot adaptability on new tasks and limiting rapid forward transfer.

Additionally, As shown in Section 4.1, large-scale pretraining on over 1k heterogeneous robotic tasks may yield weaker performance on simpler tasks than training from scratch. This may stem from the

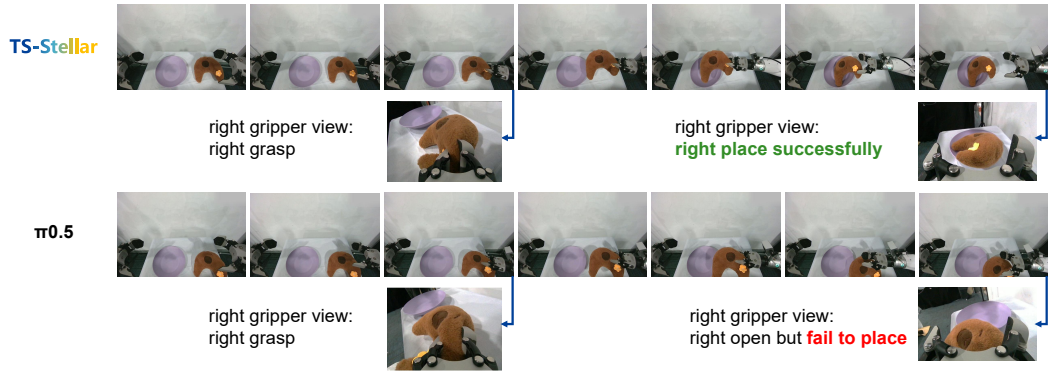


Figure 14: **Behavior visualization on "Pick Toy Place Plate" after training on "Pull Stick from Bag".** TS-Stellar accurately completes both the single-arm grasping and single-arm placing sub-stages, whereas $\pi_{0.5}$ [2] fails to place the toy accurately onto the plate during the placing stage.

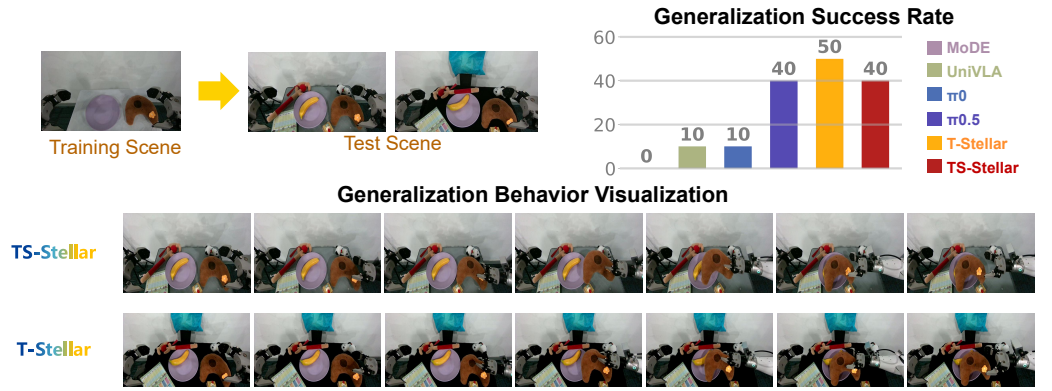


Figure 15: **Generalization Experiments on "Pick Toy Place Plate" after training on "Pull Stick from Bag".** We evaluate background generalization in cluttered scenes containing diverse objects placed on different tabletop environments. Both TS-Stellar and T-Stellar maintain successful target grasping and placing at certain locations, exhibiting moderate generalization ability.

limited capacity of the single 10-dimensional task knowledge representation to capture task diversity, resulting in excessive expert overlap and increased task interference, as discussed in Section F.1. Nevertheless, Stellar VLA still outperforms state-of-the-art VLA baselines [1, 6, 2], highlighting the potential of structured knowledge modeling to enhance skill understanding in VLA models.

Future work will explore scaling up both the pretrained VLM backbone and the action prediction network, as well as expanding the task knowledge space to higher-dimensional and multi-token representations. These directions aim to improve performance under large-scale robotic pretraining and enable more effective continual policy generalization across diverse tasks.

Towards Task-Adaptive Knowledge Space Updates in the Early Stage of Continual Learning. As discussed in Section E.3 and Section F.1, Stellar VLA, particularly TS-Stellar, tends to over-segment the knowledge space in the early stages of continual learning, which can lead to forgetting of earlier tasks. This phenomenon may be related to the interaction between task-centric representation learning and knowledge space updates. In the early stage, the model is trained on a limited number of tasks. The VAE may therefore distribute its capacity relatively evenly across these tasks, implicitly enlarging the representation scale in the task latent space. As a result, although remaining relatively compact compared to other tasks, latents from the same task become more dispersed in absolute scale, which in turn affects clustering in the knowledge space that is sensitive to feature scale.

Future work will aim to validate this hypothesis and explore adaptive strategies that leverage the number of observed tasks to dynamically adjust representation learning and knowledge space hyper-parameters across different stages of continual learning.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The experimental results explain the mentioned addressed problems in abstract.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In Section F.2, the paper discusses the limitations of our method and the potential trade-offs.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: All results in this paper are supported by experiments and we do not include any theoretical results.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The methodology describes the necessary components, and the evaluation process and details are documented in Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code and model checkpoints are publicly available through the project page.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: We provide every detail in training and testing in the main text and Appendix D.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Yes. We report the mean and standard deviation across different training seeds for the main simulation experiment results and their visualizations.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes. We describe the GPU resources in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification:Yes, we do.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: There is no societal impact of the work performed

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite the original paper that produced the code package or dataset.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Yes.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We do not include crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: We do not include crowdsourcing experiments or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this research does not involve LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.