# Incremental Learning of Retrievable Skills For Efficient Continual Task Adaptation

**Daehee Lee**[♠,◇], **Minjong Yoo**[♠], **Woo Kyung Kim**[♠], **Wonje Choi**[♠], **Honguk Woo**[♠]
[♠]Sungkyunkwan University [◇]Carnegie Mellon University
{dulgi7245, mjyoo2, kwk2696, wjchoi1995, hwoo}@skku.edu

## Abstract

Continual Imitation Learning (CiL) involves extracting and accumulating task knowledge from demonstrations across multiple stages and tasks to achieve a multi-task policy. With recent advancements in foundation models, there has been a growing interest in adapter-based CiL approaches, where adapters are established parameter-efficiently for tasks newly demonstrated. While these approaches isolate parameters for specific tasks and tend to mitigate catastrophic forgetting, they limit knowledge sharing among different demonstrations. We introduce IsCiL, an adapter-based CiL framework that addresses this limitation of knowledge sharing by incrementally learning shareable skills from different demonstrations, thus enabling sample-efficient task adaptation using the skills particularly in non-stationary CiL environments. In IsCiL, demonstrations are mapped into the state embedding space, where proper skills can be retrieved upon input states through prototype-based memory. These retrievable skills are incrementally learned on their corresponding adapters. Our CiL experiments with complex tasks in Franka-Kitchen and Meta-World demonstrate robust performance of IsCiL in both task adaptation and sample-efficiency. We also show a simple extension of IsCiL for task unlearning scenarios.

## 1 Introduction

Lifelong agents such as home robots are required to continually adapt to new tasks in sequential decision-making situations by leveraging knowledge from past experiences. However, many real-world domains pose substantial challenges for these lifelong agents; the complexity and ever-changing nature of these tasks make it difficult for agents to constantly adapt, leading to difficulties in retaining knowledge and maintaining operational efficiency [1]. For instance, a home robot agent, operating within a single household, needs to continuously adapt, learning specific tasks in various areas such as cooking assistance in the kitchen or cleaning in the bathroom. At the same time, it is crucial that the agent not only retains but also improves its proficiency in the tasks it has previously learned, ensuring that it maintains consistent efficiency throughout the home.

For these lifelong agents, Continual Imitation Learning (CiL) has been explored, in which an agent progressively learns a series of tasks by leveraging expert demonstrations over time to achieve a multi-task policy. Yet, CiL often encounters practical challenges: (1) the high costs and inefficiencies associated with comprehensive expert demonstrations [2] that are required for imitation, (2) frequently shifting tasks in dynamic, non-stationary environments, and (3) privacy concerns [3] related to learning from expert demonstrations. In this context, CiL faces significant issues in terms of cost, adaptability, and privacy, complicating its implementation in real-world scenarios.

Corresponding author: Honguk Woo (hwoo@skku.edu). Daehee Lee is currently a visiting scholar at Carnegie Mellon University.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-0549

To address these challenges, our work focuses on incorporation of skill learning and fine-tuning in CiL, leveraging recent advancements in foundation models [4, 5]. These have been increased interests in continual task adaptation based on multiple adapters learned on a foundation model [6, 7]. The adapter-based learning approach allows for parameter isolation for individual tasks, thus enabling to mitigate catastrophic forgetting of previously learned knowledge in CiL. Motivated by this use of adapters, we develop IsCiL, a new adapter-based CiL framework that addresses the practical challenges of CiL aforementioned, by incrementally learning shareable skills from different demonstrations through multiple adapters. IsCiL facilitates sample-efficient task adaptation using the skills particularly in non-stationary CiL environments.

Specifically, in the IsCiL framework, a prototype-based skill incremental learning method is employed with a two-level hierarchy including smaller, more manageable adapters: skill retriever and skill decoder. The skill retriever is responsible for composing skills to complete given goal-reaching tasks. It utilizes skill prototypes, which are representative embeddings of skills, to retrieve the appropriate skill for input. The knowledge of each skill is contained within the adapter, which can modify its associated base model output. The skill decoder is responsible for producing short-horizon actions for state-skill pairs.

We evaluate IsCiL and several adapter-based continual learning baselines across scenario variations based on complex, long-horizon tasks in the Franka-Kitchen and Meta-World environments to assess sample efficiency, task adaptation, and privacy considerations. The baselines include adapter-based continual adaptation techniques as well as conventional continual imitation learning methods. Our results demonstrate that IsCiL achieves robust performance without requiring comprehensive expert demonstrations. This flexibility allows IsCiL to continually and efficiently adapt to varying sequences in different environments by leveraging any available expert data to learn useful skills, with tasks composed of diverse instructions and demonstrations.

In summary, the IsCiL framework enhances sample efficiency and task adaptation, effectively bridging the gap between adapter-based CiL approaches and the knowledge sharing across demonstrations. Comprehensive experiments demonstrate that IsCiL outperforms other adapter-based continual learning approaches in various CiL scenarios.

## 2 Related work

**Continual imitation learning.** To tackle the problem of catastrophic forgetting in continual learning, numerous studies have employed rehearsal techniques [8, 9, 10, 11], which involve replaying past experiences to maintain performance on previously learned tasks. Another approach involves utilizing additional model parameters to progressively extend the model architecture [12, 13, 14, 15, 16]. These methods adapt the model's structure over time to accommodate new tasks. However, rehearsal techniques exhibit high variability in forgetting depending on the replay ratio and often demand substantial training to incorporate new knowledge [17]. Progressive models, on the other hand, require stage identification during evaluation and often overlook unseen tasks [13]. In this work, we propose a CiL framework that enables effective learning and expansion without requiring rehearsal and stage identification, leveraging pre-trained goal-based model knowledge.

**Continual task adaptation with pre-trained models.** Several recent works use pre-trained models, accumulating knowledge continually through additional Parameter Efficient Tuning (PET) modules such as adapters [18, 17, 19, 20, 21, 6, 22]. These methods enhance the flexibility and scalability of continual learning systems. However, they suffer from inaccurate matching between adapter selection and trained knowledge, leading to a misalignment between the knowledge learned during training and the knowledge used during evaluation [17, 20], which hinders overall performance. In the realm of sequential decision making, some studies have explored adapting pre-trained models. In [6], the state space of tasks is fully partitioned, restricting its applicability in more integrated environments. Meanwhile, [7] relies on comprehensive demonstrations for learning, which may be impractical in real-world scenarios. Our study aims to enhance task adaptation efficiency by using incrementally generalized skills with accurate matching on state space.

**Skill adaptation.** Reinforcement learning research has enhanced fast adaptation through skill exploration [23] and skill priors [24], focusing on improving sample efficiency with offline datasets. Despite these advancements, adapting fixed skill decoders to new environments remains challenging. To overcome these limitations, skill-based few-shot imitation learning methods have been developed

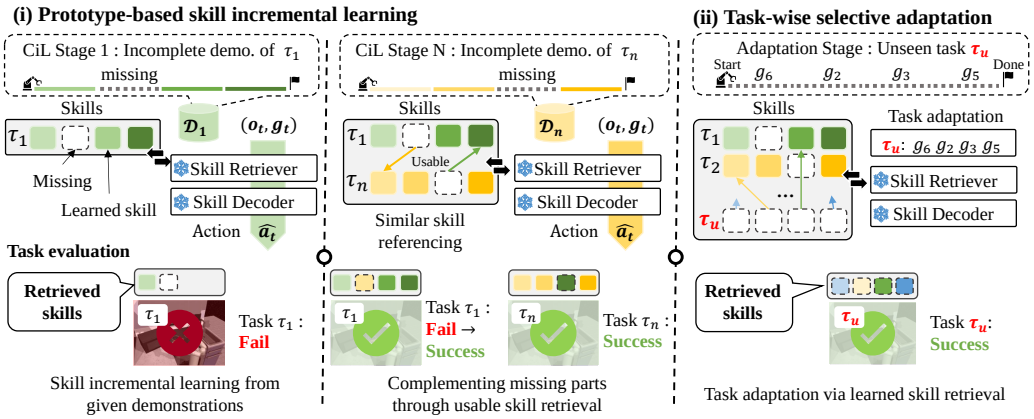


Figure 1: The scenario demonstrating how IsCiL enhances continual imitation learning efficiency through retrievable skills: (i) Prototype-based skill incremental learning: despite the failure of $\tau_1$, skills are incrementally learned from the available demonstrations. In later stages, missing skills for $\tau_1$ are retrieved from other tasks, achieving the resolution of $\tau_1$ and illustrating the reversibility and efficiency of retrievable skills. (ii) Task-wise selective adaptation: IsCiL effectively retrieves relevant learned skills, facilitating rapid task adaptation.

[25, 26]. However, these methods require extensive past data and struggle with scalability and generalization. Even skill-based approaches used in continual imitation learning [8] still require rehearsal data to mitigate knowledge loss and face difficulties addressing privacy issues through unlearning. Our IsCiL employs parameter-efficient skill adapters to prevent catastrophic forgetting and maintain efficiency, providing a scalable solution for unlearning.

## 3 Approaches

Our work addresses three key challenges of CiL: (1) data inefficiency, (2) non-stationarity, and (3) privacy concerns, by adopting retrievable skills in the CiL context. Specifically, our IsCiL framework not only enhances data-efficient continual task evaluation in a non-stationary environment but also supports unlearning as a task adaptation strategy, thereby mitigating privacy concerns.

### 3.1 Problem formulation

In CiL scenarios, we consider a data stream of task datasets $\{\mathcal{D}_i\}_{i=1}^{p}$, where $\mathcal{D}_i$ contains an expert demonstration $\mathcal{D}_i = \{d_i^1, ..., d_i^N\}$ for its associated task $\tau_i$. To effectively represent complex long-horizon tasks, each task $\tau_i$ is comprised of sub-goal list, $\tau = \{g_i^1, ..., g_i^M\}$. Each task dataset is sampled in a finite-horizon markov decision process $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mu_0, H)$, where $\mathcal{S}$ is a state space, $\mathcal{A}$ is a action space, $\mathcal{P}$ is a transition probability, $\mathcal{R}$ is a reward function, $\mu_0$ is an initial state distribution, and $H$ is an environment horizon.

For demonstration $d = \{(s_t, a_t)\}_{t=1}^{H}$, a state $s_t \in S$ represents a tuple $(o_t, g_t)$ consisting of an observation $o_t$ and a sub-goal $g_t$. In our work, we represent sub-goals through language and use language-based goal embeddings for $g_t$ to achieve language-conditioned policies. Then, the objective of IsCiL is to obtain a multi-task policy $\pi^*$, by which the performance on the tasks in the data stream can be comparable to that of respective expert policies. This is formulated as

$$\pi^* = \underset{\pi}{\operatorname{argmin}} \left[ \mathbb{E}_i \left[ \sum_{\tau \in \mathcal{T}_i} \mathrm{KL}(\pi(\cdot|s) \| \tilde{\pi}_\tau(\cdot|s)) \right] \right] \tag{1}$$

where $\tilde{\pi}_\tau$ represents an expert policy for $\tau$ and $\mathcal{T}_i$ denotes a set of evaluation tasks at stage $i$. In this context, the evaluation tasks continuously vary across different stages.


https://doi.org/10.52202/079017-0549

### 3.2 Overall architecture

To effectively handle complicated CiL scenarios, we present the IsCiL framework which involves (i) **prototype-based skill incremental learning** and (ii) **task-wise selective adaptation**.

As illustrated in Figure 1, in (i) the prototype-based skill incremental learning, we use a two-level hierarchy structure with a skill retriever $\pi_R$ composing the skills for each sub-goal, and a skill decoder $\pi_D$ producing short-horizon actions based on state-skill pairs. For this two-level policy hierarchy, we employ a skill prototype-based approach, in which skill prototypes capture the sequential patterns of actions and associated environmental states, as observed from expert demonstrations. These prototypes serve as a reference for skills learned from a multi-stage data stream. Using these skill prototypes, we can effectively translate task-specific instructions or demonstrations into a series of appropriate skills.

Through this prototype-based skill retrieval method, the policy flexibly uses skills that are shareable among tasks, potentially learned in the past or future, for policy evaluation. This enables the CiL agent to effectively learn diverse tasks and rapidly adapt to variations, while incrementally accumulating skill knowledge from a multi-stage data stream. Furthermore, to facilitate sample-efficient learning and enhance stability in CiL, we employ parameter-efficient adapters that are continually fine-tuned on a base model. Each skill knowledge is encapsulated within a dedicated adapter and incorporated into the skill decoder $\pi_D$ to infer expert actions.

In (ii) the task-wise selective adaptation, we devise efficient task adaptation procedures in the policy hierarchy to adapt to specific tasks using incrementally learned skills. This enables the CiL agent to not only facilitate adaptation to shifts in task distribution (e.g., due to non-stationary environment conditions) but also support task unlearning upon explicit user request (e.g., due to privacy concerns).

Suppose that the smart home environment undergoes an upgrade with the installation of new smart lighting systems throughout the house. In this case, task-wise selective adaptation can be used for rapid adaptation by removing outdated control routines associated with the previous systems.

### 3.3 Prototype-based skill incremental learning

**State encoder and prototype-based skill retriever.** To facilitate skill retrieval from demonstrations, we encode observation and goal pairs $(o_t, g_t)$ into state embeddings $s_t$ using a function $f : (o_t, g_t) \mapsto s_t$. We implement $f$ as a fixed function to ensure consistent retrieval results for learning efficiency, mitigating the negative effects of input distribution shifts.

To effectively handle the multi-modality of the state distribution in non-stationary environments, we employ a skill retriever $\pi_R$. For this, we use multifaceted skill prototypes $\chi_z \in \mathcal{X}$, where $\mathcal{X}$ is the set of learned skill prototypes. These prototypes capture the sequential patterns of expert demonstrations associated with specific goal-reaching tasks.

$$\theta_z = \pi_R(s_t; \mathcal{X}) = h\left(\text{argmax}_{\chi_z \in \mathcal{X}} S(\chi_z, s_t)\right), \text{ where } S(\chi_z, s_t) = \max_{b \in \chi_z} \text{sim}(b, s_t) \quad (2)$$

Here, $h : \chi_z \mapsto \theta_z$ denotes a one-to-one function that maps each skill prototype $\chi_z$ to its dedicated adapter parameters $\theta_z$, while the similarity function $S$ is defined as the maximum similarity between state $s$ and bases $b \in \chi_z$. Each $\chi_z$ consists of multiple bases (e.g., 20 bases), and each basis $b$ is a representative vector containing its corresponding centroid, shaped identically to the state $s_t$.

**Adapter conditioned skill decoder.** To effectively use the knowledge of the pre-trained base model without forgetting, even in a non-stationary changing environment, the skill decoder is conditioned based on parameters. The skill decoder policy $\pi_D(\hat{a}_t | o_t, g_t; \theta_{\text{pre}}, \theta_z)$ operates with the skill adapter parameters $\theta_z$ and the pre-trained base model $\theta_{\text{pre}}$, using the Low-Rank Adaptation [27].

**Skill incremental learning.** To incrementally learn new retrievable skills, we update the skill prototype and adapter pair $(\chi_{z^*}, \theta_{z^*})$ for a novel skill $z^*$. The skill prototype $\chi_{z^*}$ is created by dividing a dataset of a single skill into several clusters based on similarity. From each cluster, a representative value is extracted to serve as the basis $b$, representing $z^*$. We use the KMeans algorithm [28] to determine these bases, ensuring that the number of bases $|\chi_z|$ adequately captures the diversity within the dataset of the novel skill. This multifaceted set of bases allows the skill prototype to capture an accurate multi-modal distribution of the skill represented in the state space, enabling effective retrieval as described in Eq. 2. In our experiment, $z^*$ is created for each sub-goal $g$ in the given dataset $\mathcal{D}_i$ for each stage $i$.

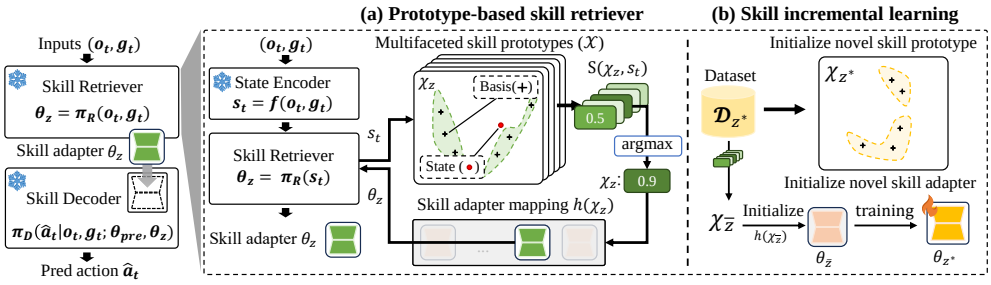


Figure 2: Overview of the IsCiL framework: (a) The prototype-based skill retriever sequentially utilizes a state encoder $f$, multifaceted skill prototypes $\mathcal{X}$, and a skill adapter mapping function $h$ to identify the skill adapter $\theta_z$. (b) Skill incremental learning involves the initialization and updating of the skill prototype $\chi_{z^*}$ and its corresponding adapter $\theta_{z^*}$.

The learning of the skill adapter is divided into two phases: initialization and update. During the initialization phase, $\theta_{z^*}$ is initialized using existing skill adapters. Predictions with the existing skill dataset and skill prototypes $\chi_z \in \mathcal{X}$ are used to identify the most frequently selected skill. Average scores are computed for each skill prototype using the dataset involved in training $z^*$, as defined in Eq. 2. The skill with the highest average score, denoted as $\bar{z}$, is selected. Consequently, $\theta_{z^*}$ is initialized by $\theta_{\bar{z}}$. Then, the initialized adapter is updated through the following imitation loss.

$$\mathcal{L}(o_t, g_t, a_t; \theta_z) = \|a - \pi_D(\hat{a}_t \mid o_t, g_t; \theta_{\text{pre}}, \theta_z)\|, \text{ where } \theta_z = \pi_R(o_t, g_t) \tag{3}$$

The novel skill $z^*$ is incorporated into the learned prototypes $\mathcal{X} \leftarrow \mathcal{X} \cup \chi_{z^*}$, and the novel prototype and adapter pair $(\chi_{z^*}, \theta_{z^*})$ updates the function $h$ for pair mapping. Figure 2 presents an overview of this methodology, along with the algorithm for incremental learning is detailed in Appendix B.1.

### 3.4 Task-wise selective adaptation

**Task evaluation.** Given the pre-trained model $\theta_{\text{pre}}$ and learned skill prototypes $\mathcal{X}$, for given inputs $(o_t, g_t)$ from the environment, IsCiL performs the following evaluation process.

$$\hat{a}_t \sim \pi_D(\hat{a}_t \mid o_t, g_t; \theta_{\text{pre}}, \theta_z), \text{ where } \theta_z = \pi_R(o_t, g_t; \mathcal{X}) \tag{4}$$

The evaluation process adapts to novel tasks and sub-goal sequences from the environment by modifying the goal $g_t$. This adjustment enables the inference of appropriate current actions, in a manner of similar to handling learned tasks. For example, a kitchen robot tailored to a specific user's kitchen setup can continuously and instantly adapt to changes in recipes without additional training.

**Task unlearning.** To ensure privacy protection for incrementally learned skills, our architecture allows for task unlearning by removing task-specific skill prototypes and adapters. In IsCiL, the separation of skill adapters for each task facilitates easy tagging of task information on each skill. When an unlearning request is given with a task identifier $\tau$, the corresponding skill prototypes and adapters are removed. This approach ensures exceptionally efficient and effective unlearning, aligning with the strong unlearning strategies in continual learning discussed in [3].

## 4 Experiments

### 4.1 Environments and data streams

To investigate the sample efficiency and adaptation performance, we construct complex CiL scenarios using diverse long-horizon tasks [29, 30, 31]. We then analyze the sample efficiency across different stages and tasks with three types of scenarios: *Complete*, *Semi*-complete, and *Incomplete*, depending on how the samples are utilized and shared. Each scenario consists of a pre-training stage followed by 20 CiL stages. Figure 3 illustrates these scenarios.

**Evolving Kitchen.** Evolving Kitchen is a data stream based on long-horizon tasks in the Franka-Kitchen environment [29, 30]. Each task requires sequentially achieving four out of seven sub-goals. The scenario consists of a pre-training stage in the environment with only four objects: kettle, bottom burner, top burner, and light switch, followed by continual adaptation to tasks involving seven objects.


https://doi.org/10.52202/079017-0549

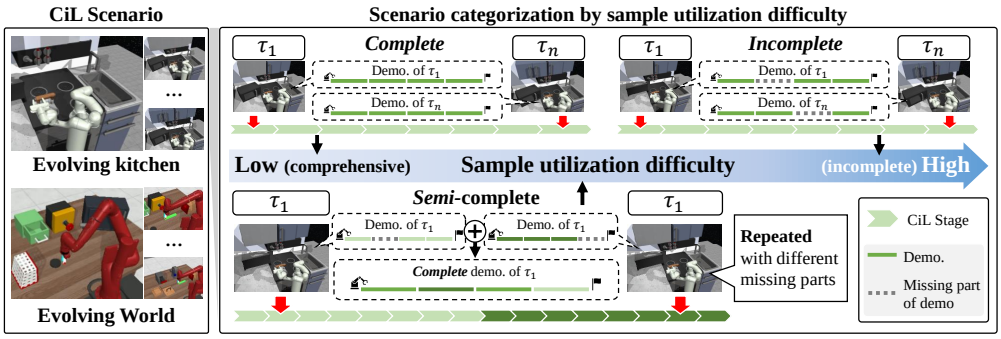


Figure 3: CiL scenarios including *Complete*, *Semi-Complete*, and *Incomplete*, categorized by sample utilization difficulty, based on the completeness of the demonstration for task performance: In *Complete*, each of the 20 CiL stages incrementally introduces new tasks featuring objects not encountered in the pre-training stage, along with full, comprehensive demonstrations for each task. In *Semi-Complete*, the first 10 stages are repeated twice, with tasks presented alongside incomplete demonstrations, where specific sub-goals are missing from the trajectories. In *Incomplete*, the same sequence of tasks from the *Complete* scenario is used, but all stages feature incomplete demonstrations, requiring the system to handle tasks with missing sub-goal trajectories.

**Evolving World.** Evolving World is a data stream based on the Meta-World environment [31] with long-horizon tasks, similar to [32, 33, 34]. Each task requires sequentially achieving four out of eight sub-goals. The scenario consists of a pre-training stage in the environment with only four objects, followed by continual adaptation to an entire environment with all eight Meta-World objects. More detailed configurations are provided in Appendix A.

### 4.2 Baselines and metrics

**Baselines.** We implement continual imitation learning and continual adaptation methods for sequential decision-making problems, which do not use rehearsal. First, we consider continual learning algorithms which involve full-model updates (**Seq**, **EWC** [35]). We also implement several continual adaptation approaches that utilize pre-trained models with adapters (**L2M** [6], **TAIL** [7]). L2M learns a key and adapter pair to modulate the pre-trained model, where the key is a retrievable state embedding similar to our prototypes. TAIL, unlike L2M, incrementally constructs task identifiers and corresponding adapters to modulate the pre-trained model with new task data without forgetting previous tasks. Each method is categorized based on the values used for adapter retrieval: a version that uses no additional identifiers, sub-goal identifiers (denoted as **-**$g$), and whole sub-goal sequences as single identifiers (denoted as **-**$\tau$). Additionally, we include a **Multi-task** learning approach as an oracle baseline, which retains all incoming data at each stage and utilizes it for training in subsequent stages. For all baselines, we use the same pre-trained goal-conditioned policy and a diffusion model [36, 37] as the base policy architecture. A detailed description of the baselines and their hyperparameters are provided in Appendix B.2.

**Metrics.** We use three metrics to report CiL performance: Forward Transfer (FWT), Backward Transfer (BWT), and Area Under Curve (AUC) [38, 7]. In our long-horizon tasks, these metrics rely on goal-conditioned success rates (GC), which measure the ratio of successfully completed sub-goals to the total sub-goals within each task [39].

- **FWT** (Forward Transfer): This evaluates the ability to learn tasks using previously learned knowledge. It is measured by the performance of a task when it occurs.
- **BWT** (Backward Transfer): This evaluates the impact of each learning stage on the performance of tasks learned in previous stages. It measures the change in task performance from past stages observed in the current stage.
- **AUC** (Area Under Curve): This represents the overall continual imitation learning performance in a scenario. It measures the average performance of tasks learned in the current stage over the remaining stages of the scenario.

For all metrics, **higher values indicate better** performance, with details provided in Appendix B.3.

https://doi.org/10.52202/079017-0549

Table 1: Overall performance on CiL scenarios of Evolving Kitchen and Evolving World: The rows represent baselines, categorized into sequential adaptation and adapter-based approaches, and oracle, respectively. The columns represent continual learning scenarios, where each scenario has 20 stages. Each scenario in the environment is categorized into *Complete*, *Semi*-complete, and *Incomplete*. The highest performance is highlighted in **bold** and the second highest performance is underlined.

| **Stream** | Evolving Kitchen-*Complete* | | | Evolving Kitchen-*Semi* | | | Evolving Kitchen-*Incomplete* | | |
|---|---|---|---|---|---|---|---|---|---|
| CiL-algorithm | FWT (%) | BWT (%) | AUC (%) | FWT (%) | BWT (%) | AUC (%) | FWT (%) | BWT (%) | AUC (%) |
| Pre-trained | - | - | 24.3±0.5 | - | - | 29.1±0.9 | - | - | 24.3±0.5 |
| Seq-FT | 90.9±2.6 | -63.7±2.7 | 35.0±0.7 | 37.1±2.1 | -25.1±2.7 | 16.5±0.7 | 32.7±4.3 | -19.6±3.0 | 15.7±0.5 |
| EWC | 34.2±0.8 | -19.5±4.2 | 17.1±2.7 | 27.2±1.3 | -18.0±1.3 | 12.2±1.4 | 19.3±2.3 | -3.2±11.3 | 10.4±1.7 |
| Seq-LoRA | 77.5±2.6 | -55.2±1.8 | 28.3±1.5 | 37.4±3.8 | -25.5±3.2 | 15.9±1.6 | 32.9±2.5 | -19.9±2.9 | 14.5±0.2 |
| L2M | 24.7±4.8 | -2.5±4.5 | 22.7±1.6 | 19.2±4.4 | 0.2±1.3 | 19.1±4.8 | 17.5±4.0 | -2.0±3.2 | 15.8±4.8 |
| L2M-$g$ | 38.2±3.4 | -6.5±3.7 | 32.3±1.4 | 37.9±3.7 | -4.5±3.1 | 32.1±1.2 | 37.5±10.0 | -6.5±6.9 | 31.0±8.8 |
| TAIL-$g$ | 85.3±8.0 | -49.9±6.7 | 41.5±1.7 | 55.0±1.5 | -21.1±2.2 | 37.2±2.4 | 53.2±1.7 | -20.0±2.0 | 35.4±0.7 |
| TAIL-$\tau$ | 86.2±5.6 | 0.0±0.0 | 86.2±5.6 | 41.2±2.5 | 0.0±0.0 | 41.2±2.5 | 33.8±3.0 | 0.0±0.0 | 33.8±3.0 |
| IsCiL (ours) | 79.3±1.7 | 11.0±1.6 | **89.8±0.5** | 68.1±2.2 | 8.6±0.6 | **75.8±1.8** | 61.8±0.9 | 13.7±2.9 | **74.0±1.9** |
| Multi-task | 93.3±1.7 | -1.6±2.3 | 92.3±1.8 | 75.4±4.5 | 8.0±5.5 | 83.2±1.1 | 71.7±1.1 | 12.6±0.8 | 83.0±1.1 |

| **Stream** | Evolving World-*Complete* | | | Evolving World-*Semi* | | | Evolving World-*Incomplete* | | |
|---|---|---|---|---|---|---|---|---|---|
| CiL-algorithm | FWT (%) | BWT (%) | AUC (%) | FWT (%) | BWT (%) | AUC (%) | FWT (%) | BWT (%) | AUC (%) |
| Pre-trained | - | - | 0.0±0.0 | - | - | 0.0±0.0 | - | - | 0.0±0.0 |
| Seq-FT | 88.9±3.1 | -73.6±4.2 | 24.9±0.4 | 38.9±5.9 | -27.5±5.5 | 13.2±0.9 | 41.4±2.0 | -33.0±2.0 | 12.2±0.8 |
| EWC | 25.7±3.8 | -18.0±0.2 | 10.5±3.5 | 13.9±1.4 | -9.1±1.8 | 6.2±1.8 | 18.2±2.8 | -11.6±2.1 | 8.5±0.9 |
| Seq-LoRA | 85.6±2.9 | -75.1±2.3 | 21.4±1.2 | 32.2±5.2 | -18.2±4.9 | 16.0±2.3 | 38.1±1.6 | -30.6±0.9 | 11.7±0.9 |
| L2M | 72.1±5.3 | -6.6±2.1 | 65.9±3.3 | 41.0±2.1 | 6.3±3.0 | 47.0±0.7 | 26.1±1.1 | 5.7±2.8 | 31.4±2.0 |
| L2M-$g$ | 64.2±3.9 | -19.3±4.4 | 48.6±2.0 | 44.5±2.0 | 3.4±2.5 | 48.2±0.2 | 33.2±2.0 | -0.6±0.9 | 33.1±2.2 |
| TAIL-$g$ | 90.0±3.0 | -56.8±0.4 | 39.5±2.9 | 43.2±7.8 | -17.6±3.5 | 27.4±5.1 | 51.4±2.5 | -21.4±0.6 | 32.5±2.3 |
| TAIL-$\tau$ | 85.7±5.9 | 0.0±0.0 | **85.7±5.9** | 27.5±0.7 | 0.0±0.0 | 27.5±0.7 | 39.7±1.0 | 0.0±0.0 | 39.7±1.0 |
| IsCiL (ours) | 81.7±0.4 | 2.7±0.9 | 84.3±1.1 | 60.0±1.1 | 9.3±1.4 | **68.9±0.5** | 63.2±1.5 | 8.7±2.7 | **71.2±4.2** |
| Multi-task | 88.6±3.6 | 2.8±3.5 | 90.7±1.2 | 55.0±3.6 | 27.6±4.1 | 80.9±0.3 | 73.2±1.7 | 12.6±1.2 | 84.2±1.3 |

### 4.3 Overall performance : sample efficiency

Table 1 shows the CiL performance on Evolving Kitchen and Evolving World across three different scenarios (*Complete*, *Semi*, *Incomplete*). We compare the performance achieved by our framework IsCiL and other baselines (L2M, TAIL) with different conditioning values ($g$,$\tau$) for adapter retrieval. IsCiL consistently demonstrates superior performance in AUC across all scenarios, achieving between 84.5% and 97.2% of the oracle baseline (Multi-task learning). TAIL-$\tau$ shows the most competitive performance in the *Complete* CiL scenario across both environments. However, due to its isolated adapter for learning and evaluation, it fails to effectively utilize samples across stages.

L2M and L2M-$g$ exhibit relatively lower and less stable AUC in the Evolving Kitchen scenario. Conversely, in Evolving World-*Semi*, they surpass TAIL-$\tau$ in AUC. This demonstrates that they are capable of sharing different skills across stages. Despite this, they still struggle with accurately retrieving the correct skill or suffer from performance degradation due to knowledge overwriting. Unlike them, IsCiL effectively mitigates overwriting by maintaining distinct skill representations across stages. Both L2M-$g$ and TAIL-$g$, which aim to leverage sub-goal labels for CiL, struggle to maintain performance due to skill distribution shifts, leading to catastrophic forgetting of skills for sub-goals. These challenges reveal that relying solely on sub-goal labels may not be sufficient to sustain and share skills effectively across different stages and tasks.

Both Seq-FT and Seq-LoRA struggle with forgetting. This is evident in the *Complete* scenario, where Seq-FT achieves the highest FWT but shows the lowest BWT, leading to a decline in overall performance. EWC exhibits consistently lower performance, as the regularization used to preserve past knowledge significantly hinders learning on current tasks, leading to severe degradation in long-horizon tasks. Although EWC shows higher BWT compared to other sequential tuning baselines, its low FWT limits overall effectiveness.


https://doi.org/10.52202/079017-0549

Table 2: Task adaptation performance with unseen tasks: This is based on the existing Evolving World-*Complete* and Evolving Kitchen-*Complete*. In Evolving World, four novel tasks are introduced every four stages, while in Evolving Kitchen, two novel tasks are introduced every five stages. Metrics with the suffix -A denote performance based solely on adaptation tasks, while other metrics report performance across all tasks.

| **Stream** | Evolving Kitchen-*Complete* Unseen | | | | | Evolving World-*Complete* Unseen | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Algorithm | FWT (%) | BWT (%) | AUC (%) | FWT-A (%) | AUC-A (%) | FWT (%) | BWT (%) | AUC (%) | FWT-A (%) | AUC-A (%) |
| Seq-FT | 72.3±1.6 | -47.7±1.6 | 30.4±0.2 | 27.8±0.6 | 19.5±0.1 | 52.9±3.6 | -26.7±1.8 | 30.1±2.1 | 16.3±1.8 | 24.0±2.6 |
| EWC | 21.0±15.9 | -14.0±2.0 | 16.8±1.6 | 18.1±4.2 | 14.4±1.6 | 16.5±1.9 | -8.1±0.8 | 9.6±2.6 | 6.1±1.3 | 8.3±2.1 |
| Seq-LoRA | 62.4±3.8 | -41.5±3.3 | 25.4±0.9 | 28.1±0.0 | 18.2±0.0 | 45.2±0.4 | -35.8±1.3 | 14.5±0.9 | 6.4±2.5 | 8.2±1.8 |
| L2M | 22.3±2.3 | 0.3±1.5 | 22.7±3.5 | 15.3±3.2 | 21.2±4.1 | 55.1±3.7 | -1.4±3.3 | 53.6±1.0 | 40.3±2.4 | 41.2±2.0 |
| L2M-$g$ | 33.8±0.9 | -4.3±1.2 | 30.0±0.4 | 22.2±0.6 | 24.1±0.7 | 43.3±1.6 | -8.2±3.6 | 35.7±1.6 | 24.2±1.5 | 25.7±1.7 |
| TAIL-$g$ | 67.6±7.4 | -34.9±5.4 | 36.8±3.2 | 34.7±2.2 | 30.1±1.0 | 53.2±1.4 | -27.1±1.2 | 29.2±0.3 | 18.6±0.9 | 19.1±0.6 |
| IsCiL (ours) | 69.5±2.5 | 16.3±2.2 | **84.4**±1.3 | **52.1**±7.5 | **72.8**±2.1 | 64.3±2.6 | -0.5±3.5 | **63.9**±0.6 | **45.8**±4.7 | **45.3**±0.9 |
| Multi-task | 85.3±1.7 | 3.7±1.8 | 88.8±0.0 | 70.8±0.0 | 79.0±0.1 | 85.4±0.9 | 5.6±0.5 | 90.4±0.5 | 78.3±2.9 | 85.9±0.4 |

Table 3: Overall performance with task unlearning as task adaptation: Additional stages for unlearning tasks that were learned during other stages are included for tests.

| **Stream** | Evolving Kitchen-*Complete* Unlearning | | | Evolving Kitchen-*Incomplete* Unlearning | | |
|---|---|---|---|---|---|---|
| Algorithm | FWT (%) | BWT (%) | AUC (%) | FWT (%) | BWT (%) | AUC (%) |
| TAIL-$\tau$ CLPU | 86.2±5.6 | 0.0±0.0 | 86.2±5.6 | 33.8±3.0 | 0.0±0.0 | 33.8±3.0 |
| IsCiL (ours) | 75.0±7.2 | 11.2±5.5 | 85.2±1.8 | 61.4±2.9 | 12.4±2.9 | 72.7±2.9 |

### 4.4 Task adaptation

Table 2 shows the unseen task adaptation ability of IsCiL, where only the sub-goal sequence of novel task is provided without demonstrations. This scenario extends the existing *Complete* CiL scenarios by periodically introducing novel tasks. Metrics labeled with the suffix -A indicate results from adaptation tasks, whereas the other metrics reflect performance on all tasks. For this scenario, we exclude TAIL-$\tau$ from comparison, as it lacks the ability to adapt to novel tasks.

IsCiL demonstrates superior task performance in both scenarios, which contributes to greater efficiency in task adaptation. Moreover, in Evolving Kitchen, IsCiL not only demonstrates task adaptation ability by achieving the highest FWT-A, but also significantly enhances its initial performance, raising FWT-A from 52.1 to an AUC-A of 72.8. TAIL-$g$ shows comparable performance in FWT for the Evolving Kitchen. However, it struggles with catastrophic forgetting, leading to a $-34.9$ negative BWT when faced with significant distribution shifts in sub-goal demonstrations. In Evolving World, L2M, which actively learns to share skills during training, outperforms TAIL-$g$. L2M is the only baseline achieving performance improvement on unseen tasks through CiL.

### 4.5 Task unlearning as adaptation

Table 3 measures CiL performance in scenario with task-level unlearning. For comparison, we use an adapter-based approach with parameter isolation-based continual learning private unlearning (CLPU) [3], extending TAIL to **TAIL-$\tau$ CLPU** and IsCiL without skill adapter initialization. Similar to IsCiL, CLPU learns tasks in isolated models tagged with specific task identifiers and handles unlearning requests by removing the corresponding model parameters of the target task. Both **TAIL-$\tau$ CLPU** and IsCiL ensure output distribution equality between the unlearned model and the model trained with the retained dataset. Thus, their CiL performance remains largely unaffected by unlearning.

Although IsCiL exhibits a slight performance degradation of $1.8\% \sim 5.2\%$ after unlearning, as reported in Table 1, it still demonstrates robustness by achieving a $115\%$ higher AUC compared to **TAIL-$\tau$ CLPU** in *incomplete* scenarios.

### 4.6 Analysis

**Rehearsal comparison.** Figure 4 compares the sample efficiency to retain learned knowledge between IsCiL and a rehearsal-based continual imitation learning approach, Experience Replay (ER) [40]. For ER, we adjust the number of stored samples per learning stage, while IsCiL does not store rehearsals

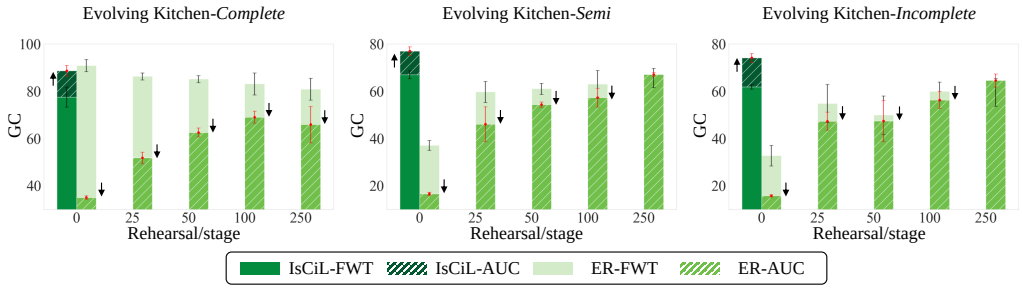


Figure 4: Comparison w.r.t. the number of rehearsals: The horizontal axis represents the amount of stored rehearsal data at each stage, while the vertical axis indicates goal-conditioned success rates (GC).

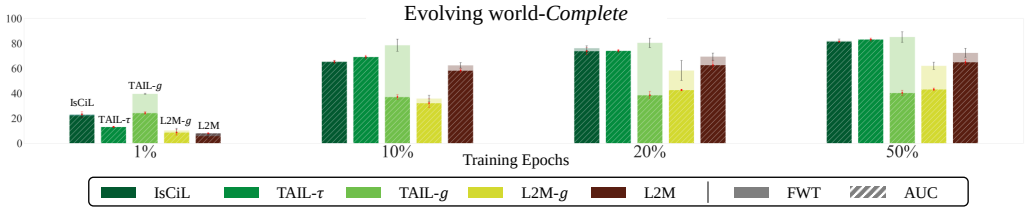


Figure 5: Comparison w.r.t. training resources: In all baselines, the plain bar graph represents FWT, while the bar graph with hatch marks represents AUC. The vertical axis indicates goal-conditioned success rates (GC).

for training. IsCiL achieves the highest AUC in all environments and is the only approach where AUC surpasses FWT. ER shows comparable FWT in *Complete*, but as the number of stored samples increases, FWT decreases, indicating that more rehearsals actually reduce training sample efficiency. In *Semi* and *Incomplete*, using 250 rehearsals (approximately 5% of the stage dataset) yields FWT comparable to IsCiL but rarely improves AUC.

**Limited training resource.** Figure 5 shows the computational efficiency of IsCiL in resource-constrained training settings, as discussed in [38]. In this experiment, the training resources is limited to 1% to 50% of those used in Table 1. IsCiL and TAIL-$\tau$ show robust performance for varied training resources. TAIL-$g$ shows higher FWT, as it trains the same sub-goal data on the same adapter, which excels in learning new tasks, but it fails to retain that knowledge. However, using skill data from different stages to update the same adapter makes it vulnerable to skill distribution shifts in CiL; this ends up with significant AUC degradation.

### 4.7 Ablation

Table 4 investigates the impact of the number of prototype bases on CiL performance, showing that increasing the number of bases improves both AUC and result stability, particularly around K=10. Results are reported based on units ($g$ and $\tau$) used to construct new skill prototypes and the corresponding number of bases. IsCiL with a single base fails to effectively learn task knowledge, achieving similar performance to L2M in Table 1, due to insufficient representation of the skill distribution. Additionally, the IsCiL framework maintains positive BWT scores, demonstrating its ability to leverage future samples to enhance past performance. IsCiL with $\tau$, which constructs new skills based on entire task trajectories, required more bases in proportion to the increase in the number of transitions involved in constructing the skill trajectory to maintain stability.

Table 4: Ablation on IsCiL skill prototype

| **Stream** | Evolving kitchen-*Complete* | | |
|---|---|---|---|
| Ablations | FWT (%) | BWT (%) | AUC (%) |
| IsCiL $g, \lvert\chi_z\rvert = 20$ | 79.3±1.7 | 11.0±1.6 | 89.8±0.5 |
| IsCiL $g, \lvert\chi_z\rvert = 1$ | 28.9±10.2 | 2.3±5.7 | 30.6±12.2 |
| IsCiL $g, \lvert\chi_z\rvert = 5$ | 63.1±4.0 | 2.7±7.3 | 66.5±7.9 |
| IsCiL $g, \lvert\chi_z\rvert = 10$ | 76.4±6.5 | 8.2±4.0 | 83.9±2.7 |
| IsCiL $g, \lvert\chi_z\rvert = 25$ | 77.1±1.8 | 11.9±1.7 | 88.2±1.1 |
| IsCiL $g, \lvert\chi_z\rvert = 50$ | 81.5±2.8 | 7.9±4.9 | 89.4±1.2 |
| IsCiL $\tau, \lvert\chi_z\rvert = 20$ | 57.8±16.6 | 10.9±1.8 | 67.2±17.2 |
| IsCiL $\tau, \lvert\chi_z\rvert = 80$ | 84.3±6.7 | 5.0±7.5 | 89.5±8.3 |


https://doi.org/10.52202/079017-0549

## 5 Conclusion

In this study, we presented the IsCiL framework to address key challenges in continual imitation learning (CiL). Our approach incorporates adapter-based skill learning, leveraging multifaceted skill prototypes and an adapter pool to effectively capture the distribution of skills for continual task adaptation. IsCiL specifies enhanced sample efficiency and robust task adaptation, effectively bridging the gap between adapter-based CiL approaches and the need for knowledge sharing across staged demonstrations. Comprehensive experiments demonstrate that IsCiL consistently outperforms other adapter-based continual learning approaches in various CiL scenarios.

**Limitations.** Like other adapter-based CiL approaches, IsCiL requires extra computation for evaluation, which can create overhead, especially in resource-constrained environments. It also depends on sub-goal sequences for training and evaluation, adding complexity and resource demands. Another limitation is determining the appropriate size of the adapter parameters, which depends on the performance of the pre-trained base model and the degree of task shift, making optimal adaptation challenging. Moreover, balancing the stability of the embedding function with the prototype size remains an area that requires further refinement to achieve optimal performance.

## Acknowledgement

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) RS-2022-II220043 (2022-0-00043), Adaptive Personality for Intelligent Agents, RS-2022-II221045 (2022-0-01045), Self-directed multi-modal Intelligence for solving unknown, open domain problems, RS-2019-II190421, Artificial Intelligence Graduate School Program (Sungkyunkwan University), the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2023-00213118) and by Samsung Electronics.

## References

[1] Byeonghwi Kim, Minhyuk Seo, and Jonghyun Choi. Online continual learning for interactive instruction following agents. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[2] Suneel Belkhale, Yuchen Cui, and Dorsa Sadigh. Data quality in imitation learning. In *Advances in neural information processing systems (NeurIPS)*, 2023.

[3] Bo Liu, Qiang Liu, and Peter Stone. Continual learning and private unlearning. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2022.

[4] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems (NeurIPS)*, 2021.

[5] Kuang-Huei Lee, Ofir Nachum, Mengjiao Sherry Yang, Lisa Lee, Daniel Freeman, Sergio Guadarrama, Ian Fischer, Winnie Xu, Eric Jang, Henryk Michalewski, et al. Multi-game decision transformers. *Advances in neural information processing systems (NeurIPS)*, 2022.

[6] Thomas Schmied, Markus Hofmarcher, Fabian Paischer, Razvan Pascanu, and Sepp Hochreiter. Learning to modulate pre-trained models in rl. *Advances in neural information processing systems (NeurIPS)*, 36, 2024.

[7] Zuxin Liu, Jesse Zhang, Kavosh Asadi, Yao Liu, Ding Zhao, Shoham Sabach, and Rasool Fakoor. TAIL: Task-specific adapters for imitation learning with large pretrained models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[8] 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)*, 2024.

[9] Philemon Schopf, Sayantan Auddy, Jakob Hollenstein, and Antonio Rodriguez-Sanchez. Hypernetwork-ppo for continual reinforcement learning. In *Deep RL Workshop at NeurIPS*, 2022.

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

[11] Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine. Why generalization in rl is difficult: Epistemic pomdps and implicit partial observability. *Advances in neural information processing systems (NeurIPS)*, 2021.

[12] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[13] 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 (CVPR)*, 2018.

[14] Xilai Li, Yingbo Zhou, Tianfu Wu, Richard Socher, and Caiming Xiong. Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting. In *International Conference on Machine Learning (ICML)*, 2019.

[15] Dushyant Rao, Francesco Visin, Andrei Rusu, Razvan Pascanu, Yee Whye Teh, and Raia Hadsell. Continual unsupervised representation learning. *Advances in neural information processing systems (NeurIPS)*, 2019.

[16] Tiantian Zhang, Zichuan Lin, Yuxing Wang, Deheng Ye, Qiang Fu, Wei Yang, Xueqian Wang, Bin Liang, Bo Yuan, and Xiu Li. Dynamics-adaptive continual reinforcement learning via progressive contextualization. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.

[17] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. *European Conference on Computer Vision (ECCV)*, 2022.

[18] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[19] Qiankun Gao, Chen Zhao, Yifan Sun, Teng Xi, Gang Zhang, Bernard Ghanem, and Jian Zhang. A unified continual learning framework with general parameter-efficient tuning. *International Conference on Computer Vision (ICCV)*, 2023.

[20] James Seale Smith, Leonid Karlinsky, Vyshnavi Gutta, Paola Cascante-Bonilla, Donghyun Kim, Assaf Arbelle, Rameswar Panda, Rogerio Feris, and Zsolt Kira. Coda-prompt: Continual decomposed attention-based prompting for rehearsal-free continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[21] Wei-Cheng Huang, Chun-Fu Chen, and Hsiang Hsu. OVOR: Oneprompt with virtual outlier regularization for rehearsal-free class-incremental learning. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

[22] Martin Wistuba, Prabhu Teja Sivaprasad, Lukas Balles, and Giovanni Zappella. Continual learning with low rank adaptation. In *NeurIPS 2023 Workshop on Distribution Shifts (DistShifts)*, 2023.

[23] Seohong Park, Kimin Lee, Youngwoon Lee, and Pieter Abbeel. Controllability-aware unsupervised skill discovery. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, 2023.

[24] Karl Pertsch, Youngwoon Lee, and Joseph J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning (CoRL)*, 2020.

[25] Kourosh Hakhamaneshi, Ruihan Zhao, Albert Zhan, Pieter Abbeel, and Michael Laskin. Hierarchical few-shot imitation with skill transition models. In *International Conference on Learning Representations (ICLR)*, 2022.

[26] Soroush Nasiriany, Tian Gao, Ajay Mandlekar, and Yuke Zhu. Learning and retrieval from prior data for skill-based imitation learning. In *Conference on robot learning (CoRL)*, 2022.

[27] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*, 2022.

[28] S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 1982.

[29] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

[30] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

[31] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning (CoRL)*, 2020.

 https://doi.org/10.52202/079017-0549

[32] Suraj Nair, Eric Mitchell, Kevin Chen, Silvio Savarese, Chelsea Finn, et al. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on robot learning (CoRL)*, 2022.

[33] Divyansh Garg, Skanda Vaidyanath, Kuno Kim, Jiaming Song, and Stefano Ermon. Lisa: Learning interpretable skill abstractions from language. *Advances in neural information processing systems (NeurIPS)*, 2022.

[34] Sangwoo Shin, Daehee Lee, Minjong Yoo, Woo Kyung Kim, and Honguk Woo. One-shot imitation in a non-stationary environment via multi-modal skill. In *International Conference on Machine Learning (ICML)*, 2023.

[35] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 2017.

[36] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in neural information processing systems (NeurIPS)*, 2020.

[37] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. In *International Conference on Learning Representations (ICLR) 11*, 2023.

[38] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, qiang liu, Yuke Zhu, and Peter Stone. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2023.

[39] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[40] 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.

[41] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 2017.

# A Environment and Data Stream Details

## A.1 Franka Kitchen

We conduct experiments on the Franka kitchen environment [29, 30]. Each Franka kitchen task comprises of 4 sub-goals, from total pool of 7: microwave, kettle, bottom burner, top burner, light switch, slide cabinet, and hinge cabinet. Observation is a 60-dimensional vector, which is a combination of the positions and velocities of 7-DoF robot arm and interacting objects. We express sub-goal information using language embedding. The target sub-goal of the current state is acquired by a pre-defined environmental reward and task, and a sub-goal sequence to solve. We use 24 tasks in the 'mixed' dataset from the D4RL [29]. In the pre-training stage, we train the model only on tasks comprised of following four sub-goals: kettle, bottom burner, top burner, light switch.

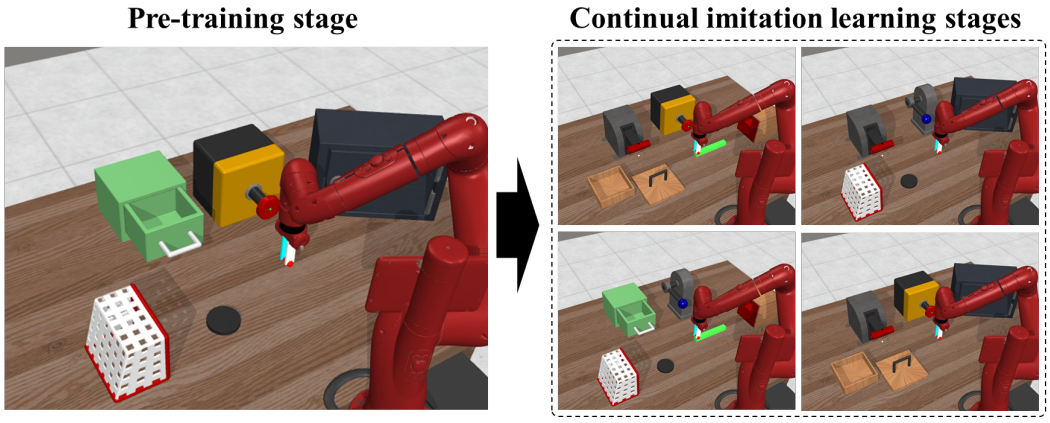


Figure 6: Example of a multi-stage Meta-World environment in our continual imitation learning scenarios.

## A.2 Multi-stage Meta World

We conduct experiments on the multi-stage variation of the Meta-World environment [31, 34]. Each Meta-World task comprises of 4 sub-goals from total pool of 8: puck, box, handle, drawer, lever, button, door, and stick. The environments are divided into different scenarios based on which 4 out of 8 objects are placed on the table. For each environment, tasks are defined according to the sequence in which the 4 sub-goals must be achieved. Observation is a 140-dimensional vector, which contains the positions and velocities of the 4-DoF robot arm and all interacting objects in the environment. In this environment, sub-goal information is also expressed using language embedding. The expert dataset is collected using a heuristic expert policy provided by Meta-World [31]. In the pre-training stage of Meta World, we train the model on 24 tasks on a environment consisting of four objects: a puck, a drawer, a button, and a door.

## A.3 Data Stream

**Evolving Kitchen** Tables 5 and 6 display the detailed configurations of the Evolving Kitchen in our CiL scenario. Each task involves sequentially solving its respective sub-goals. The underlined sub-goals (e.g., kettle) are those missing in the *Semi Complete* and *Incomplete* scenarios.

**Evolving World** Tables 7 and 8 display the detailed configurations of our Evolving World CiL scenario. Similarly, the Evolving World is also presented in the same way as the Evolving Kitchen.

**Unseen Task Adaptation** Tables 9 and 10 show the detailed configurations of unseen tasks for our Evolving Kitchen-*Complete* Unseen and Evolving World-*Complete* Unseen. In the Evolving Kitchen, 2 new tasks appear every 5 stages. In the Evolving World, 4 new tasks appear every 4 stages. Each new task includes only the sequence of sub-goals that must be completed in order, without any demonstrations.


https://doi.org/10.52202/079017-0549

Table 5: Evolving Kitchen-*Complete & Incomplete* data stream task configuration

| Evolving Kitchen-*Complete & Incomplete* | | | | |
|---|---|---|---|---|
| **Task** | **Sub-goal 1** | **Sub-goal 2** | **Sub-goal 3** | **Sub-goal 4** |
| $\tau_1$ | microwave | kettle | top burner | light switch |
| $\tau_2$ | kettle | bottom burner | top burner | slide cabinet |
| $\tau_3$ | microwave | bottom burner | top burner | slide cabinet |
| $\tau_4$ | kettle | bottom burner | light switch | slide cabinet |
| $\tau_5$ | microwave | kettle | light switch | slide cabinet |
| $\tau_6$ | kettle | bottom burner | top burner | hinge cabinet |
| $\tau_7$ | microwave | kettle | top burner | hinge cabinet |
| $\tau_8$ | microwave | kettle | slide cabinet | hinge cabinet |
| $\tau_9$ | kettle | light switch | slide cabinet | hinge cabinet |
| $\tau_{10}$ | microwave | kettle | bottom burner | hinge cabinet |
| $\tau_{11}$ | kettle | bottom burner | slide cabinet | hinge cabinet |
| $\tau_{12}$ | kettle | bottom burner | light switch | hinge cabinet |
| $\tau_{13}$ | microwave | top burner | light switch | hinge cabinet |
| $\tau_{14}$ | microwave | kettle | bottom burner | slide cabinet |
| $\tau_{15}$ | microwave | kettle | light switch | hinge cabinet |
| $\tau_{16}$ | microwave | bottom burner | top burner | light switch |
| $\tau_{17}$ | kettle | top burner | light switch | slide cabinet |
| $\tau_{18}$ | microwave | bottom burner | top burner | hinge cabinet |
| $\tau_{19}$ | microwave | bottom burner | slide cabinet | hinge cabinet |
| $\tau_{20}$ | microwave | bottom burner | light switch | slide cabinet |

Table 6: Evolving Kitchen-*Semi* data stream task configuration

| Evolving Kitchen-*Semi* | | | | |
|---|---|---|---|---|
| **Task** | **Sub-goal 1** | **Sub-goal 2** | **Sub-goal 3** | **Sub-goal 4** |
| $\tau_1$ | microwave | kettle | top burner | light switch |
| $\tau_2$ | kettle | bottom burner | top burner | slide cabinet |
| $\tau_3$ | microwave | bottom burner | top burner | slide cabinet |
| $\tau_4$ | kettle | bottom burner | light switch | slide cabinet |
| $\tau_5$ | microwave | kettle | light switch | slide cabinet |
| $\tau_6$ | kettle | bottom burner | top burner | hinge cabinet |
| $\tau_7$ | microwave | kettle | top burner | hinge cabinet |
| $\tau_8$ | microwave | kettle | slide cabinet | hinge cabinet |
| $\tau_9$ | kettle | light switch | slide cabinet | hinge cabinet |
| $\tau_{10}$ | microwave | kettle | bottom burner | hinge cabinet |
| $\tau_{11}$ | microwave | kettle | top burner | light switch |
| $\tau_{12}$ | kettle | bottom burner | top burner | slide cabinet |
| $\tau_{13}$ | microwave | bottom burner | top burner | slide cabinet |
| $\tau_{14}$ | kettle | bottom burner | light switch | slide cabinet |
| $\tau_{15}$ | microwave | kettle | light switch | slide cabinet |
| $\tau_{16}$ | kettle | bottom burner | top burner | hinge cabinet |
| $\tau_{17}$ | microwave | kettle | top burner | hinge cabinet |
| $\tau_{18}$ | microwave | kettle | slide cabinet | hinge cabinet |
| $\tau_{19}$ | kettle | light switch | slide cabinet | hinge cabinet |
| $\tau_{20}$ | microwave | kettle | bottom burner | hinge cabinet |

Table 7: Evolving World-*Complete & Incomplete* data stream task configuration

| Evolving World-*Complete & Incomplete* | | | | |
|---|---|---|---|---|
| **Task** | **Sub-goal 1** | **Sub-goal 2** | **Sub-goal 3** | **Sub-goal 4** |
| $\tau_1$ | door | handle | button | box |
| $\tau_2$ | puck | drawer | stick | lever |
| $\tau_3$ | handle | puck | lever | door |
| $\tau_4$ | button | drawer | box | stick |
| $\tau_5$ | door | handle | box | button |
| $\tau_6$ | lever | stick | drawer | puck |
| $\tau_7$ | lever | puck | handle | door |
| $\tau_8$ | stick | button | drawer | box |
| $\tau_9$ | handle | button | box | door |
| $\tau_{10}$ | drawer | stick | lever | puck |
| $\tau_{11}$ | puck | lever | door | handle |
| $\tau_{12}$ | stick | button | box | drawer |
| $\tau_{13}$ | handle | button | door | box |
| $\tau_{14}$ | drawer | lever | stick | puck |
| $\tau_{15}$ | puck | lever | handle | door |
| $\tau_{16}$ | stick | box | button | drawer |
| $\tau_{17}$ | handle | door | box | button |
| $\tau_{18}$ | stick | drawer | puck | lever |
| $\tau_{19}$ | door | puck | lever | handle |
| $\tau_{20}$ | box | drawer | button | stick |

Table 8: Evolving World-*Semi* data stream task configuration

| Evolving World-*Semi* | | | | |
|---|---|---|---|---|
| **Task** | **Sub-goal 1** | **Sub-goal 2** | **Sub-goal 3** | **Sub-goal 4** |
| $\tau_1$ | door | handle | button | box |
| $\tau_2$ | puck | drawer | stick | lever |
| $\tau_3$ | handle | puck | lever | door |
| $\tau_4$ | button | drawer | box | stick |
| $\tau_5$ | door | handle | box | button |
| $\tau_6$ | lever | stick | drawer | puck |
| $\tau_7$ | lever | puck | handle | door |
| $\tau_8$ | stick | button | drawer | box |
| $\tau_9$ | handle | button | box | door |
| $\tau_{10}$ | drawer | stick | lever | puck |
| $\tau_{11}$ | door | handle | button | box |
| $\tau_{12}$ | puck | drawer | stick | lever |
| $\tau_{13}$ | handle | puck | lever | door |
| $\tau_{14}$ | button | drawer | box | stick |
| $\tau_{15}$ | door | handle | box | button |
| $\tau_{16}$ | lever | stick | drawer | puck |
| $\tau_{17}$ | lever | puck | handle | door |
| $\tau_{18}$ | stick | button | drawer | box |
| $\tau_{19}$ | handle | button | box | door |
| $\tau_{20}$ | drawer | stick | lever | puck |

**Unlearning Scenario** In the Unlearning Scenario, 1 learned task is unlearned every 5 learning stages. In the Evolving Kitchen Unlearning scenario, $\tau_4$, $\tau_8$, $\tau_{13}$, and $\tau_{17}$ are sequentially unlearned.

# B Experiment Details

## B.1 IsCiL Implementation

IsCiL consists of two modules: a skill retriever, $\pi_R$, and a skill decoder, $\pi_D$. The skill retriever $\pi_R$ includes three components: a state encoder $f$, skill prototypes $\mathcal{X}$, and a skill adapter mapping function $h$. Each skill prototype $\chi_z$ in $\mathcal{X}$ is composed of 20 bases $b$. To modulate skill decoder $\pi_D$, we use Low Rank Adaptation(LoRA) [27]. In our experiment, we used 4-rank LoRA adapters for skill adapter. IsCiL training and evaluation process follows :

Table 9: Evolving Kitchen-*Complete* Unseen task configuration

| Evolving World-*Complete* Unseen | | | | |
|---|---|---|---|---|
| **Stage** | **Sub-goal 1** | **Sub-goal 2** | **Sub-goal 3** | **Sub-goal 4** |
| 5 | microwave | kettle | top burner | slide cabinet |
| 5 | microwave | kettle | top burner | top burner |
| 10 | microwave | kettle | top burner | light switch |
| 10 | kettle | bottom burner | top burner | light switch |
| 15 | kettle | top burner | slide cabinet | hinge cabinet |
| 15 | microwave | top burner | slide cabinet | hinge cabinet |
| 20 | microwave | top burner | light switch | slide cabinet |
| 20 | kettle | top burner | light switch | hinge cabinet |

Table 10: Evolving World-*Complete* Unseen task configuration

| Evolving World-*Complete* Unseen | | | | |
|---|---|---|---|---|
| **Stage** | **Sub-goal 1** | **Sub-goal 2** | **Sub-goal 3** | **Sub-goal 4** |
| 4 | door | handle | button | box |
| 4 | puck | drawer | stick | lever |
| 4 | handle | puck | lever | door |
| 4 | button | drawer | box | stick |
| 8 | door | handle | box | button |
| 8 | puck | lever | drawer | stick |
| 8 | handle | lever | puck | door |
| 8 | box | drawer | stick | button |
| 12 | box | handle | door | button |
| 12 | lever | drawer | stick | puck |
| 12 | handle | lever | puck | door |
| 12 | box | drawer | stick | button |
| 16 | door | handle | box | button |
| 16 | puck | drawer | stick | lever |
| 16 | handle | puck | lever | door |
| 16 | box | drawer | stick | button |
| 20 | door | handle | box | button |
| 20 | lever | drawer | stick | puck |
| 20 | door | handle | lever | puck |
| 20 | box | drawer | stick | button |

**Algorithm 1** IsCiL Skill Incremental Learning

State encoding function $f$, Skill retriever $\pi_R$
Skill decoder $\pi_D$, Pre-trained parameter $\theta_{\text{pre}}$
Skill adapter mapping function $h$
**for** each stage $i$ in CiL Stages **do**
  **for** each sub-goal $g$ in task dataset $D_i$ **do**
    $D_i^g \leftarrow \{(o, g') \in D_i \mid g' = g\}$ // filter transitions related to the current sub-goal $g$
    $S_i^g \leftarrow \{f(o_t, g_t) \mid (o_t, g_t) \in D_i^g\}$ // encode states from the filtered dataset into state embeddings
    $\mathcal{X}^g \leftarrow \{\text{argmax}_{\chi_z \in \mathcal{X}} S(\chi_z, s_t) \mid s_t \in S_i^g\}$ // retrieve the most relevant skill prototypes for each state $s_t$
    $\chi_{\bar{z}} \leftarrow \text{Mode}(\mathcal{X}^g)$ // select the most frequently retrieved skill prototype from the set
    $\theta_{z^*} \leftarrow h(\chi_{\bar{z}})$ // map the selected skill prototype $\chi_{\bar{z}}$ to its skill adapter via $h$
    Update $\theta_{z^*}$ using Eq. (3) // update the skill adapter based on task-specific learning
    $\mathcal{X} \leftarrow \mathcal{X} \cup \chi_{z^*}$ // append the new skill prototype to the skill set for future retrieval
    Update the mapping function $h$ to map $\chi_{z^*}$ to the updated adapter $\theta_{z^*}$ // update $h$ with the new skill adapter
  **end for**
**end for**

**Algorithm 2** IsCiL Evaluation

State encoding function $f$, Skill retriever $\pi_R$
Skill decoder $\pi_D$, Pre-trained parameter $\theta_{\text{pre}}$
**while** not done **do**
  $s_t = f(o_t, g_t)$ // encode state
  $\theta_z = \pi_R(s_t)$ // retrieve skill
  $\hat{a}_t \sim \pi_D(o_t, g_t; \theta_{\text{pre}}, \theta_z)$ // decode the skill
**end while**


https://doi.org/10.52202/079017-0549

### B.2 Baselines

**Seq-FT & Seq-LoRA** Sequential Fine Tuning(Seq-FT) is a method that updates the entire model sequentially. The variation, Seq-LoRA, is used to determine how effectively the fixed pre-trained model can utilize its knowledge. Due to poor performance at very low ranks, Seq-LoRA was trained with a 64-rank adapter in our environment.

**EWC**[41] Elastic Weight Consolidation (EWC) regularizes the weight update by using the Fisher information matrix for each network parameter. For our experiment, we adopted the online version of EWC, which updates the Fisher information at each stage by exponential moving average, following the methods in [38, 7].We use the hyperparameter alpha, set to 10, to determine the regularization strength. For updating the online Fisher information matrix$\bar{F}_i$, we use the Fisher information matrix calculated at the current stage and apply the following formula for regularization: $\bar{F}_i = \gamma F_{i-1} + (1-\gamma)F_i$, where $\gamma$ is set to 0.9.

**L2M**[6] L2M is an adapter-based continual learning method consisting of keys and their corresponding adapters. When an input is provided, L2M uses a similarity function to search for the key corresponding to that input. Input is converted to a query and utilized to search for a key, where key is a vector with the same shape as the query. Each key is then updated to maximize its similarity to the data point associated with it. Finally, the data is used to update the adapter that corresponds to the key value found through that data. This method maximizes the diversity of key usage frequency by adjusting the similarity between keys and input values during the training phase for learning new tasks [18, 6]. In our implementation, we use the normalized state value as the input query to find the key in L2M. For L2M-$g$, we use the normalized embedding of the state concatenated with the given conditioned sub-goal information directly as a query. Our adapter pool consists of 100 adapters, each being a 4-rank LoRA adapter.

**TAIL**[7] TAIL directly assigns an adapter to the given task using the task's identifier. We directly map the given identifier to the corresponding adapter. TAIL-$g$ uses a 4-rank adapter, while TAIL-$\tau$ uses a 16-rank adapter.

**Multi-task** At each stage, the model learns from the given data and stores all data for the next stage. The data stored in the buffer is mixed with the data from each stage in a 1:1 ratio for training the model.

**ER**[40] Experience Replay (ER), similarly, retains knowledge by storing a subset of the current stage's data for the next stage. The data stored in the buffer is mixed with the data from each stage in a 1:1 ratio for training the model.

**CLPU**[3] Continual Learning Private Unlearning (CLPU) [3] is a method for managing continual learning and unlearning. In a continual learning scenario, tasks that require maintenance are trained using existing models, while data that may require unlearning is trained on independent model parameters, tagged with when and through which task each model was trained. When an unlearning request for a specific task or training stage is received, the corresponding model parameters are completely removed to eliminate the influence of the target unlearning task from the model. CLPU provides highly efficient and powerful unlearning performance with a single delete operation for tasks learned in a continual learning scenario. In our experiment, we integrate the unlearning approach CLPU with TAIL-$\tau$, which conducts training through task information-based searches, to handle unlearning requests in continual imitation learning as a comparative method.

### B.3 Metric

We report 3 metrics for CiL performance for tasks: Forward Transfer(FWT), Backward Transfer(BWT), Area Under Curve(AUC) [38, 7]. In multi-stage environment task, we report performance using the goal-conditioned success rates (GC), which evaluate the average success rate of successfully completed sub-goals out of $N$ sub-goals in the task.

- **FWT**: $\text{FWT}_\tau = \frac{1}{|I_\tau|}\sum_{i\in I_\tau} C_{\tau,i}$ where $\tau$ is task and $C_{\tau,i}$ represents the GC score of task $\tau$ at stage $i$. and $I_\tau$ is set of stage indices where task $\tau$ is trained in the CiL scenario.
- **BWT**: $\text{BWT}_\tau = \frac{1}{|I_\tau|}\sum_{i\in I_\tau}\left(\frac{1}{p-i-1}\sum_{j=i+1}^{p}(C_{\tau,j} - C_{\tau,i})\right)$, where $p$ is the final stage at which task $\tau$ is available. In the case where $p = i$ BWT is 0.

- **AUC**: $\text{AUC}_\tau = \frac{1}{|I_\tau|}\sum_{i\in I_\tau}\left(\frac{1}{p-i}\sum_{j=i}^{p} C_{\tau,j}\right)$, represent the the overall performance of continual learning, internally including FWT and BWT. In the case where $p = i$, $\text{AUC}_\tau$ is $\text{FWT}_\tau$.

The final reported metric is the average across all tasks $\tau \in \mathcal{T}$. For all metrics, higher values indicate better performance.

### B.4 Scenario training details

**Pre-trained Base Model and Stage Settings** Table 11 shows the hyperparameters and the architecture of the model we used as the base model for all baselines. Table 12 shows the common hyperparameters used to train the model for each stage in our experiments.

Table 11: Pre-trained model configure

| **Hyperparameter** | Value |
|---|---|
| Diffusion Model | DDPM [36] |
| Denoising step | 128 |
| Schedule | Linear |
| Linear start | 1e-4 |
| Linear end | 2e-2 |
| Block | MLP |
| The number of layers | 6 |
| hidden dimension | 512 |
| Layer normalization | yes |

Table 12: Continual imitation learning default hyperparameters

| **Hyperparameter** | Value |
|---|---|
| Learning rate | 5e-4 |
| Optimizer | Adam |
| Epochs/stage | 5000 |

**Pre-trained model performance** Table 13 shows the learning performance of the pre-trained model and its adaptation performance for tasks learned in scenarios without any prior training. In Evolving World, where new objects are added and the environment changes significantly, the pre-trained model failed to successfully complete any sub-goals of tasks.

Table 13: Pre-trained model performance

| **Stream**(Total phase) | Evolving Kitchen base | Evolving World-*Complete* | Evolving World-*Semi* | Evolving World-*Incomplete* |
|---|---|---|---|---|
| Evolving Kitchen Pre-trained | 98.8$\pm$0.0 | 24.3$\pm$0.5 | 29.1$\pm$0.9 | 24.3$\pm$0.5 |
| **Stream**(Total phase) | Evolving World Base | Evolving Kitchen-*Complete* | Evolving Kitchen-*Semi* | Evolving Kitchen-*Incomplete* |
| Evolving World Pre-trained | 100.0$\pm$0.0 | 0.0$\pm$0.0 | 0.0$\pm$0.0 | 0.0$\pm$0.0 |

### B.5 Compute Resources

**Computing machine** Our experimental platform is powered by an AMD 5975wx CPU and 2x RTX 4090 GPUs. The operating system used is Ubuntu 22.04.4 LTS, with Nvidia driver version 535.171.04 and CUDA version 12.2.

**Software Detail** We utilized jax 0.4.24, jaxlib 0.4.19, and flax 0.8.2 for our implementation.

**Training time** In the context of the Evolving Kitchen, each scenario involves training with three different seeds. The training duration averages 2 minutes per stage, with each stage consisting of 5000 epochs. Each scenario comprises 20 such stages, culminating in a total training time of 2 hours for a single experiment.

## C Additional Experiments

### C.1 Main experiment extension

**Training curve**. Figure 7 shows training curves of Evolving-kitchen *Complete* and *Incomplete* on Table 1. The curves provide a clear illustration of the performance progression of IsCiL and baseline methods, making changes in key metrics over the course of training easily observable.


https://doi.org/10.52202/079017-0549

### C.2 Analysis

**Skill adapter rank**. Table 14 shows the results of the ablation study on the performance of CiL based on the rank of the skill adapter. Overall, the 1-rank adapter in Evolving Kitchen demonstrates sufficient, or even superior, adaptation performance. However, in Evolving World, the 1-rank adapter leads to lower overall performance, indicating that some skills cannot be fully learned with a 1-rank adapter, resulting in a decline in performance.

**Skill decoder pre-trained model quality**. Table 15 shows the results of the ablation study on performance changes based on the quality of the pre-trained model (skill decoder). The quality of the pre-trained model varies with the number of objects included in the tasks used to pre-train the model. A decrease in the quality of the pre-trained model leads to a performance drop in both TAIL-$\tau$ and IsCiL, as the number of objects is reduced from 4 to 1.

**Scenario task sequence variation**. Table 16 shows the results of the task sequence variation analysis. We report the average performance for four different task sequences in Evolving Kitchen-*Complete*. The performance of all tasks at the final stage is not significantly affected. Since TAIL-$\tau$ learns independently for each task ID, there was no performance change with different sequences, and IsCiL also showed similar performance, indicating that task sequence variation had minimal impact on overall outcomes.

**Computational efficiency**. In our framework, skill retrieval and adaptation occur at each time step. Despite this continuous process, the impact on inference time and computational demands is minimal. Through our implementation on JAX, we observed that factors like compile optimization had a more significant effect on performance than model size. As a result, IsCiL demonstrates fast evaluation times, with retrieval and adaptation processes taking 3.6ms and 3.0ms, respectively, which ensures that IsCiL remains highly efficient during inference.

Additionally, the memory overhead required for the adapted model is minimal, with the skill adaptation adding only 0.37% to 1.48% additional parameters compared to the pre-trained model, depending on the LoRA rank (1 to 4). For skill retrieval, the parameter size of each skill prototype is relatively small, accounting for approximately 0.3% of the total model size. Furthermore, the inclusion of adapters in the skill decoder only increases the FLOPs by 3.13% of the pre-trained model, demonstrating that the retrieval and adaptation processes are computationally efficient and have a negligible impact on resource consumption.

Table 14: Ablation study on the skill adapter rank in Evolving Kitchen-*Complete* and Evolving World-*Complete*.

| **Stream** | | Evolving Kitchen-*Complete* | | | Evolving World-*Complete* | | |
|---|---|---|---|---|---|---|---|
| Rank | CiL-algorithm | FWT (%) | BWT (%) | AUC (%) | FWT (%) | BWT (%) | AUC (%) |
| 1 | L2M-$g$ | 30.2±2.1 | 2.6±1.0 | 33.0±1.6 | 56.8±3.5 | -16.9±5.2 | 41.6±1.3 |
| | TAIL-$g$ | 93.2±2.5 | -54.3±1.6 | 45.7±1.3 | 77.0±5.0 | -47.9±1.9 | 34.6±3.7 |
| | IsCiL | 89.2±4.0 | 2.7±3.0 | **91.6**±1.8 | 73.6±5.1 | -3.3±3.9 | **70.9**±3.3 |
| 4 | L2M-$g$ | 38.2±3.4 | -6.5±3.7 | 32.3±1.4 | 64.2±3.9 | -19.3±4.4 | 48.6±2.0 |
| | TAIL-$g$ | 85.3±8.0 | -49.9±6.7 | 41.5±1.7 | 90.0±3.0 | -56.8±0.4 | 39.5±2.9 |
| | IsCiL | 79.3±1.7 | 11.0±1.6 | **89.8**±0.5 | 81.7±0.4 | 2.7±0.9 | **84.3**±1.1 |

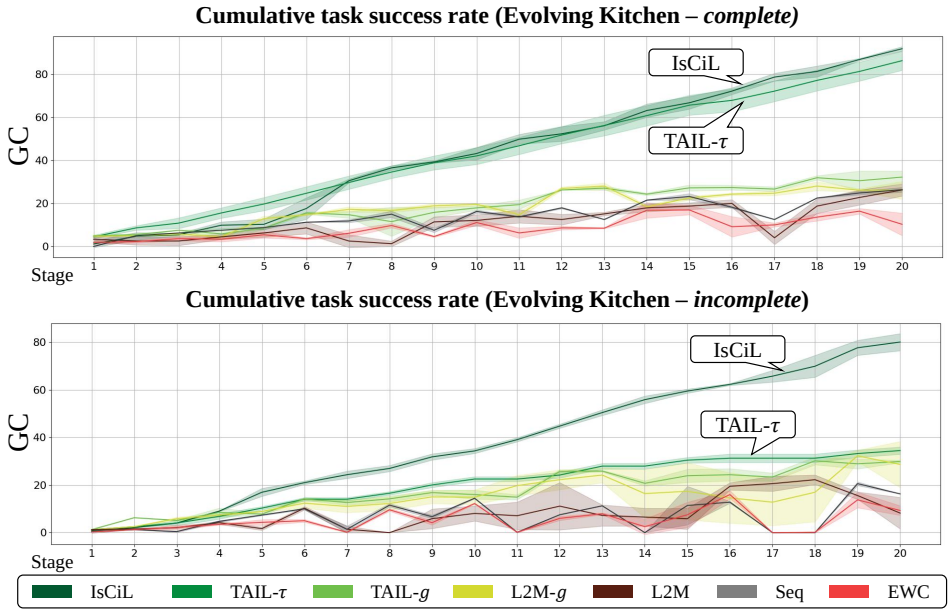


Figure 7: Evolving Kitchen-*complete* and Evolving Kitchen-*incomplete* training curves represent the cumulative task success rate up to a given stage. The goal conditioned success rate(GC) is scaled such that achieving success in all tasks by the final stage is represented as 100%. This result corresponds to the data presented in Table 1.

Table 15: Ablation study on the quality of the skill decoder pre-trained model in Evolving Kitchen-*Complete* and *Incomplete*.

| **Stream** | | Evolving Kitchen-*Complete* | | | Evolving Kitchen-*Incomplete* | | |
|---|---|---|---|---|---|---|---|
| CiL-algorithm | Pre-training | FWT (%) | BWT (%) | AUC (%) | FWT (%) | BWT (%) | AUC (%) |
| TAIL-$\tau$ | 1 object | 72.8±7.9 | 0.0±0.0 | 72.8±7.9 | 28.8±0.7 | 0.0±0.0 | 28.8±0.7 |
| | 2 object | 87.2±4.6 | 0.0±0.0 | 87.2±4.6 | 35.9±2.6 | 0.0±0.0 | 35.9±2.6 |
| | 4 object | 86.2±5.6 | 0.0±0.0 | 86.2±5.6 | 33.8±3.0 | 0.0±0.0 | 33.8±3.0 |
| IsCiL | 1 object | 60.0±4.0 | 2.1±4.2 | 62.1±0.8 | 42.1±7.3 | 5.4±3.2 | 47.0±4.6 |
| | 2 object | 78.9±5.1 | 6.4±3.7 | 84.9±1.3 | 56.7±3.2 | 12.0±2.3 | 67.3±1.6 |
| | 4 object | 79.3±1.7 | 11.0±1.6 | 89.8±0.5 | 61.8±0.9 | 13.7±2.9 | 74.0±1.9 |

Table 16: Analysis of task sequence variation in the CiL scenario of Evolving Kitchen-*Complete*.

| **Stream** | | Evolving Kitchen-*Complete* | | |
|---|---|---|---|---|
| CiL-algorithm | Task sequence | FWT (%) | BWT (%) | AUC (%) |
| IsCiL | Seq. 1 | 79.3±1.7 | 11.0±1.6 | 89.8±0.5 |
| | Seq. 2 | 80.4±2.9 | 4.4±2.7 | 87.6±1.9 |
| | Seq. 3 | 63.5±3.1 | 13.0±3.2 | 76.0±4.8 |
| | Seq. 4 | 89.6±1.9 | 1.3±1.2 | 90.8±0.9 |
| IsCiL | Average | 78.2±10.0 | 7.4±5.3 | 86.1±6.6 |
| TAIL-$\tau$ | Average | 86.2±5.6 | 0.0±0.0 | 86.2±5.6 |


https://doi.org/10.52202/079017-0549

### C.3 Scalability

**LIBERO**. Figure 8 provides a comprehensive visualization of the Skill Retriever in the LIBERO-goal scenario. The visualization highlights how the retriever successfully identifies and shares skills across different stages of CiL. This demonstrates its adaptability in handling varied states and tasks, showing its potential effectiveness even in complex LIBERO environments.

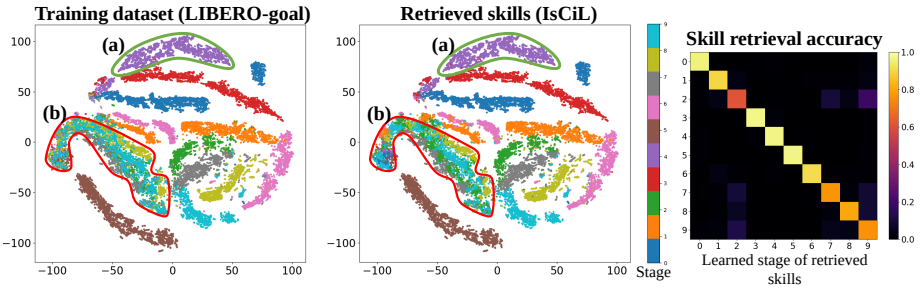


Figure 8: **Visualization of Skill Retriever on the LIBERO-goal Scenario. Left**: T-SNE visualization of the state space of the existing dataset for each stage. **Middle**: Visualization of the stages where skills retrieved by the Skill Retriever, after all CiL stages of learning. **Right**: Map showing the stages of each dataset and the retrieved skills. This demonstrates that the Skill Retriever can find skills capable of handling the given state, even in the LIBERO scenario. Additionally, in task-specific parts (a), it accurately retrieves the skills, and in parts showing similar behaviors (b), it shares skills.

## NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and precede the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

**The checklist answers are an integral part of your paper submission.** They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS paper checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

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 NA 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 NA 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 the limitations section, the paper discusses the limitations of our methodology and the potential trade-offs.


https://doi.org/10.52202/079017-0549

Guidelines:

- The answer NA 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: [NA]

   Justification: [NA]

   Guidelines:

   - The answer NA 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 the appendix.

   Guidelines:

   - The answer NA 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: Yes, we provide the codes for supplementary material.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.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://nips.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).

 https://doi.org/10.52202/079017-0549

- 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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We report these information in Appendix B.4

   Guidelines:

   - The answer NA 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: Our paper reports error bars and statistical significance information in Table 1, 2, and 3 Figure 4 and 5.

   Guidelines:

   - The answer NA 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: We report these information in Appendix B.3.

   Guidelines:

   - The answer NA 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: We conforms to the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer NA 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: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA 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., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA means that the paper poses no such risks.


https://doi.org/10.52202/079017-0549

- 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 have cited the papers for the datasets and assets used. [29, 30, 31]

    Guidelines:

    - The answer NA 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: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA 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: [NA]

    Justification: [NA]

Guidelines:

- The answer NA 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: [NA]

    Justification: [NA]

    Guidelines:

    - The answer NA 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.


https://doi.org/10.52202/079017-0549