nature machine intelligence

Article

https://doi.org/10.1038/s42256-025-00983-2

Preserving and combining knowledge in robotic lifelong reinforcement learning

Received: 1 May 2024

Accepted: 5 January 2025

Published online: 5 February 2025

Check for updates

Yuan Meng[1,5], Zhenshan Bing[1,2,5] ✉, Xiangtong Yao[1,5], Kejia Chen[1], Kai Huang[3] ✉, Yang Gao[2] ✉, Fuchun Sun[4] ✉ & Alois Knoll[1]

Humans can continually accumulate knowledge and develop increasingly complex behaviours and skills throughout their lives, which is a capability known as 'lifelong learning'. Although this lifelong learning capability is considered an essential mechanism that makes up general intelligence, recent advancements in artificial intelligence predominantly excel in narrow, specialized domains and generally lack this lifelong learning capability. Here we introduce a robotic lifelong reinforcement learning framework that addresses this gap by developing a knowledge space inspired by the Bayesian non-parametric domain. In addition, we enhance the agent's semantic understanding of tasks by integrating language embeddings into the framework. Our proposed embodied agent can consistently accumulate knowledge from a continuous stream of one-time feeding tasks. Furthermore, our agent can tackle challenging real-world long-horizon tasks by combining and reapplying its acquired knowledge from the original tasks stream. The proposed framework advances our understanding of the robotic lifelong learning process and may inspire the development of more broadly applicable intelligence.

Humans show a remarkable ability for lifelong learning by consistently acquiring knowledge and adapting to new task scenarios throughout their lives. This involves the constant and incremental development of increasingly complex behaviours, recognized as a crucial mechanism for achieving general intelligence. Recent advancements in artificial intelligence have showcased agents achieving remarkable performance across a wide range of tasks[1], such as image generation[2], article writing[3] and autonomous driving[4]. However, even though current methodologies yield impressive outcomes, they primarily focus on agents specialized in narrowly distributed tasks. In contrast, untrained agents generally require more game-play experiences throughout their lifespan than humans and struggle to generalize effectively to new variations. One notable gap between machine-intelligent agents and humans is the lack of lifelong learning capability in current intelligent agents. Lifelong learning, also referred to as incremental or continual learning[5–7], addresses the challenge of asynchronously acquiring knowledge from a continuous stream of tasks while mitigating forgetting. Its primary goal is to gradually extend the accumulated knowledge and use it for ongoing learning tasks, thereby building more complicated behaviours by knowledge combination and reapplication. This study focuses on robotic lifelong reinforcement learning (LRL), a domain where reinforcement learning provides an agent–environment interaction framework that is well suited for exploring the learning process in a sequential manner. Figure 1a illustrates the training process for a general LRL agent in the robotic context. Given an infinite stream of robotic tasks, the agent continually masters the tasks one after another, consistently accumulating knowledge and skills.

For deep learning-based algorithms, the primary challenge when facing a stream of tasks is balancing the stability and plasticity[1] of the neural networks. A common issue in this context is 'catastrophic

[1]School of Computation, Information and Technology, Technical University of Munich, Garching, Germany. [2]State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China. [3]Key Laboratory of Machine Intelligence and Advanced Computing, School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China. [4]Department of Computer Science and Technology, Tsinghua University, Beijing, China. [5]These authors contributed equally: Yuan Meng, Zhenshan Bing, Xiangtong Yao. ✉e-mail: zhenshan.bing@tum.de; huangk36@mail.sysu.edu.cn; gaoy@nju.edu.cn; fcsun@tsinghua.edu.cn

https://doi.org/10.1038/s42256-025-00983-2

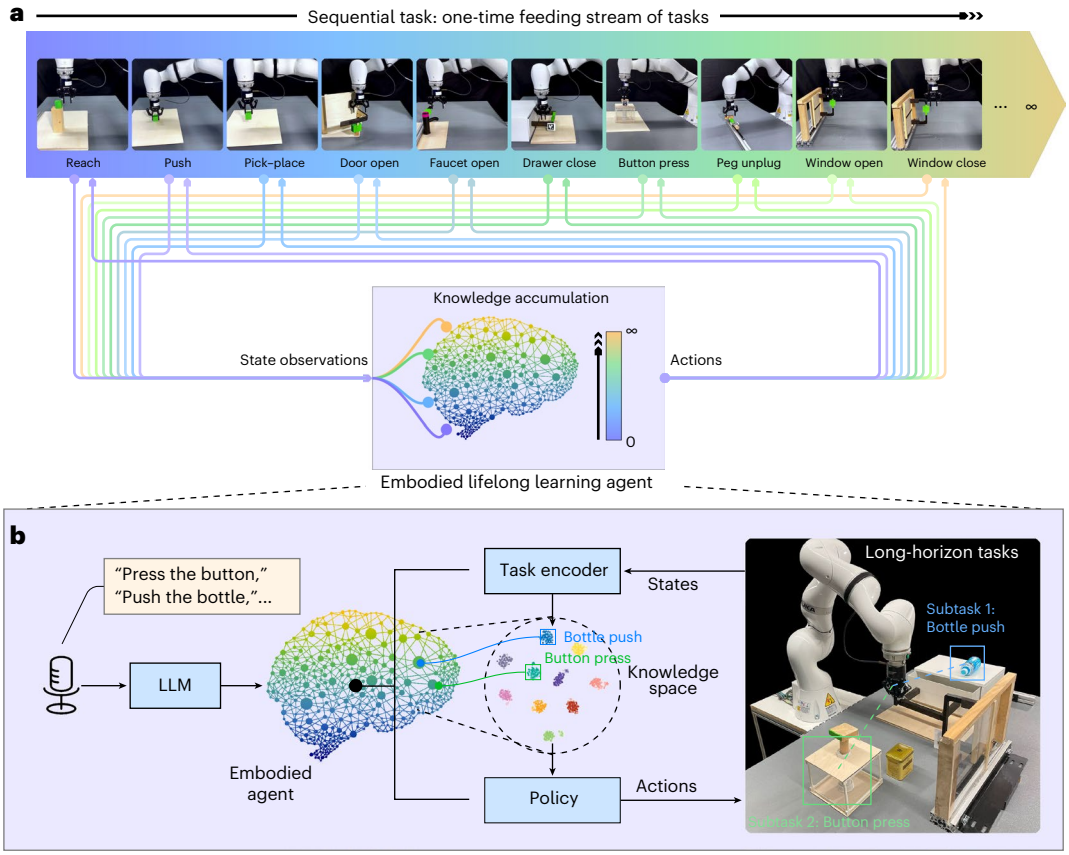


Fig. 1 | Concept illustration of robotic LRL process. a, Overview illustration of the general LRL process. Unlike the conventional multi-task approaches, where agents have simultaneous access to all tasks, an LRL agent can master tasks sequentially, one after another. Moreover, the agent should continually accumulate knowledge throughout the process. This concept emulates the human learning process. **b**, Our proposed framework under the lifelong learning concept. We instruct the deployed embodied agent to perform long-horizon tasks using language commands. The agent accomplishes these tasks through the combination and reapplication of acquired knowledge.

forgetting'[1,8]. This refers to the phenomenon where the neural network parameters associated with previously learned skills are rapidly overwritten when the agent learns new incoming tasks. Consequently, the agent's performance substantially deteriorates when revisiting previously mastered tasks. Recent lifelong machine learning studies have introduced various approaches, including regularization[5,9,10], structure modularity[11–13] and experience replay[14–16]. These methods, however, have primarily been applied to static datasets in conventional machine learning domains such as vision task classification[8,17], leaving their effectiveness in robotic learning unclear. Regularization can lead to improper parameter shifting and error accumulation, while structure modularity may struggle with dynamic adaptation when facing an unknown number of tasks. Without replay, both regularization and structure modularity methods tend to overfit on predefined tasks, lacking the flexibility to adapt to new ones in lifelong learning. Our approach does not strictly belong to any of these categories but instead draws inspiration from these methods, aiming to overcome their limitations while leveraging their strengths.

In the context of deep reinforcement learning, a common idea to avoid 'catastrophic forgetting' is through multi-task reinforcement learning (MTRL)[18–21]. In MTRL, the agent has simultaneous access to all tasks during training, avoiding the forgetting problem inherent in deep neural networks. Recent works in this domain include contextual attention-based representation learning[22], soft modularization[21], feature-wise linear modulation[23] and other baselines[19,20,24]. However, the MTRL deviates from actual human learning patterns. While MTRL tries to avoid the issue of catastrophic forgetting by providing data from various tasks simultaneously, this problem still persists in the sequential learning process. Furthermore, it relies on a predefined range of task distributions, which are often limited in scope and struggle to generalize when encountering novel non-parametric task variability[18]. Such variability shows qualitative distinction and cannot be adequately described by continuous parameters, as they require models to learn entirely new sets of rules and interactions, thus challenging their generalization capabilities[18]. Motivated by MTRL, another set of approaches to tackle the stability–plasticity dilemma is known as 'learning to learn' or meta reinforcement learning[1]. Recent studies have provided diverse approaches that enable agents to acquire knowledge across various task distributions while adapting to new tasks based on acquired knowledge in few-shot or zero-shot manner[25–29]. One notable example is continuous environment meta-reinforcement learning[25], which incorporates a Gaussian mixture model in its prior space of task encoder, which can infer and cluster the task latent representation at a meta level. However, the Gaussian mixture model faces limitations due to its reliance on a predetermined task amount, an assumption incompatible with the typically unknown or infinite task amounts in LRL.

Our study aims to develop a deep reinforcement learning framework for robotic lifelong learning. The focus is on continually learning and preserving knowledge from a stream of one-time feeding task scenarios. The proposed agent shall not forget the knowledge it acquired and can consistently perform stably on corresponding tasks throughout its lifespan. Moreover, our framework is designed to handle more complex long-horizon tasks by effectively combining and

reapplying the underlying knowledge acquired from the ongoing task stream. This highlights its capability for incremental development of progressively sophisticated behaviours. To achieve the objective, we develop a framework inspired by the Dirichlet process mixture model (DPMM), a prominent model in the Bayesian non-parametric domain, with a memoized variational Bayes inference method (memoVB)[30]. This combination enables simultaneous task inference and asynchronous knowledge preservation at the upstream level. Moreover, our framework utilizes natural language-based side information to assist in task inference. This information is encoded by a pre-trained large language model (LLM)[31]. The resulting language embedding offers the agent richer contextual insights into the current task scenario, contributing to more precise and disentangled representations in the knowledge space. As a result, the collaborative efforts of DPMM and language embeddings contribute to more accurate downstream action pattern learning. Furthermore, our embodied agent shows the ability to solve challenging, long-horizon manipulation tasks in the real world by combining and reapplying knowledge acquired throughout its lifelong learning. This showcases its potential for achieving general intelligence and may inspire the development of more broadly applicable intelligent agents. We name our proposed framework as LEGION: a Language Embedding-based Generative Incremental Off-policy Reinforcement Learning Framework with Non-parametric Bayes.

Results

In this section, we present the test results of our LEGION framework. We begin by demonstrating its performance in real-world manipulation tasks, covering both long-horizon tasks and the original sequence of single-task training. Next, we assess how knowledge is preserved in the prior space. In addition, we provide quantitative data to evaluate key aspects of LRL within our framework. Finally, we highlight the contribution of our non-parametric knowledge space in few-shot knowledge recall. The experimental set-up for both simulation and real-world experiments is detailed in 'Training and deployment' in Methods and Supplementary Section 5.

Manipulation performance

Long-horizon tasks. The deployment set-up of our framework is illustrated in Fig. 1b. To provide human commands of the task descriptions, we use a speech-recognition device and a pre-trained LLM. The trained embodied agent receives the state observations conditioned with language embeddings as inputs. After receiving the observations, the task encoder infers the knowledge to which it should apply. Subsequently, the downstream policy generates corresponding actions to accomplish the task. In the real-world scenario, we employ a KUKA iiwa robot arm as our embodiment and use a global RealSense camera to acquire vision information. A real-world video demonstration (Supplementary Video 1) showcases our embodied agent successfully completing the long-horizon task 'clean the table', which consists of seven sequential subtasks. Our agent accomplishes this by recombining the underlying knowledge gained from the one-time feeding task stream (Fig. 2), illustrating its effective generalization in the face of diverse and challenging task distributions. This ability mirrors the human learning process over a lifetime and is regarded as a key mechanism underlying general intelligence. Conventional approaches to such long-horizon tasks involve relying on human demonstrations for direct imitation. However, these approaches often result in limited generalization and flexibility when confronting varied task distributions and sequences. In contrast, our framework offers flexibility in task execution order, allowing the agent to complete the entire task in any sequence through the combination and reapplication of acquired knowledge. To highlight the generalization and flexibility of our proposed framework, we reorder the subtasks randomly and present two additional demonstrations in Supplementary Video 2. As our broad task assumption includes long-horizon tasks with strict subtask conditions as a subset, we also demonstrate how our agent solves a conventional strictly conditioned long-horizon task, 'make the coffee' (Supplementary Section 3.2).

Stream of tasks. Given a stream of one-time feeding tasks, our proposed LRL agent can master the task continually, one after the other, without forgetting previously acquired knowledge. This incremental learning approach mimics the natural human learning process and has the potential to replace, and eventually surpass, inefficient manual services in real-world applications. To assess the lifelong learning capability of our proposed agent, we implement ten distinct robotic manipulation tasks to build up a task stream. Our agent can gain knowledge asynchronously from this stream and eventually achieve the given long-horizon task (Supplementary Video 1). The agent undergoes training on each task for 1 million steps before switching to the next task. The task sequence follows an easy-to-hard task ordering (Supplementary Section 3.1): 'reach → push → pick–place → door open → faucet open → drawer close → button press → peg unplug → window open → window close'.

To demonstrate the performance of our proposed framework on its original sequence task distributions, we showcase the snapshots of its real-world tasks in Fig. 2, and we provide Supplementary Video 3 of all tasks. As observed in the snapshots and video, our proposed embodied agent completes all tasks within the given time steps. In addition, in each real-world task, we conduct at least three trials, varying the initial object positions and goal positions. The average success rates for these trials are presented in Extended Data Table 1. To demonstrate stability and robustness within the given base task distributions, our embodied agent consistently accomplishes various manipulation tasks, including 'reach', 'faucet open', 'drawer close', 'button press' and 'window open/close', leveraging asynchronously acquired knowledge. For some more challenging tasks such as 'push', 'pick–place' and 'door open', our agent can also maintain a high success rate with a score of at least 0.67.

Knowledge preservation

We evaluate knowledge preservation through t-distributed stochastic neighbor embedding (t-SNE) visualizations for intuitive understanding and statistical analysis for quantitative performance assessment during training. Furthermore, a detailed ablation study highlighting the contributions of our Bayesian non-parametric knowledge space and language embeddings is provided in Supplementary Section 3.3.

Visualization. In our framework, the task encoder initially infers the state inputs and generates the latent samples as inference results. Subsequently, the inferred task results are fitted into the non-parametric knowledge space. To assess how the acquired knowledge is preserved and managed in its space, we use t-SNE to visualize our knowledge space in a two-dimensional plane. Figure 3a–e shows the projections of knowledge space after training on two, four, six, eight and all ten tasks, respectively. Each coloured group signifies a complete task trajectory and is assigned to a cluster component in our non-parametric knowledge space. In addition, the order of these samples is represented by corresponding colour opacity, progressing from light to dark. Notably, our proposed DPMM module in the knowledge space can generate new components to store new task inference results when switching environments, facilitating the capability to infer and store new knowledge. In addition, to evaluate how our knowledge space handles acquired knowledge, we make the agent undertake the training loop twice. During the second loop, the agent revisits previously mastered tasks, whose knowledge has been preserved in the agent's knowledge space. In this phase, the agent is expected to directly utilize the existing knowledge to complete the tasks, rather than inferring a new task knowledge cluster in its prior space. We present the results after the first loop (circle markers) and after the second loop (cross markers)

https://doi.org/10.1038/s42256-025-00983-2

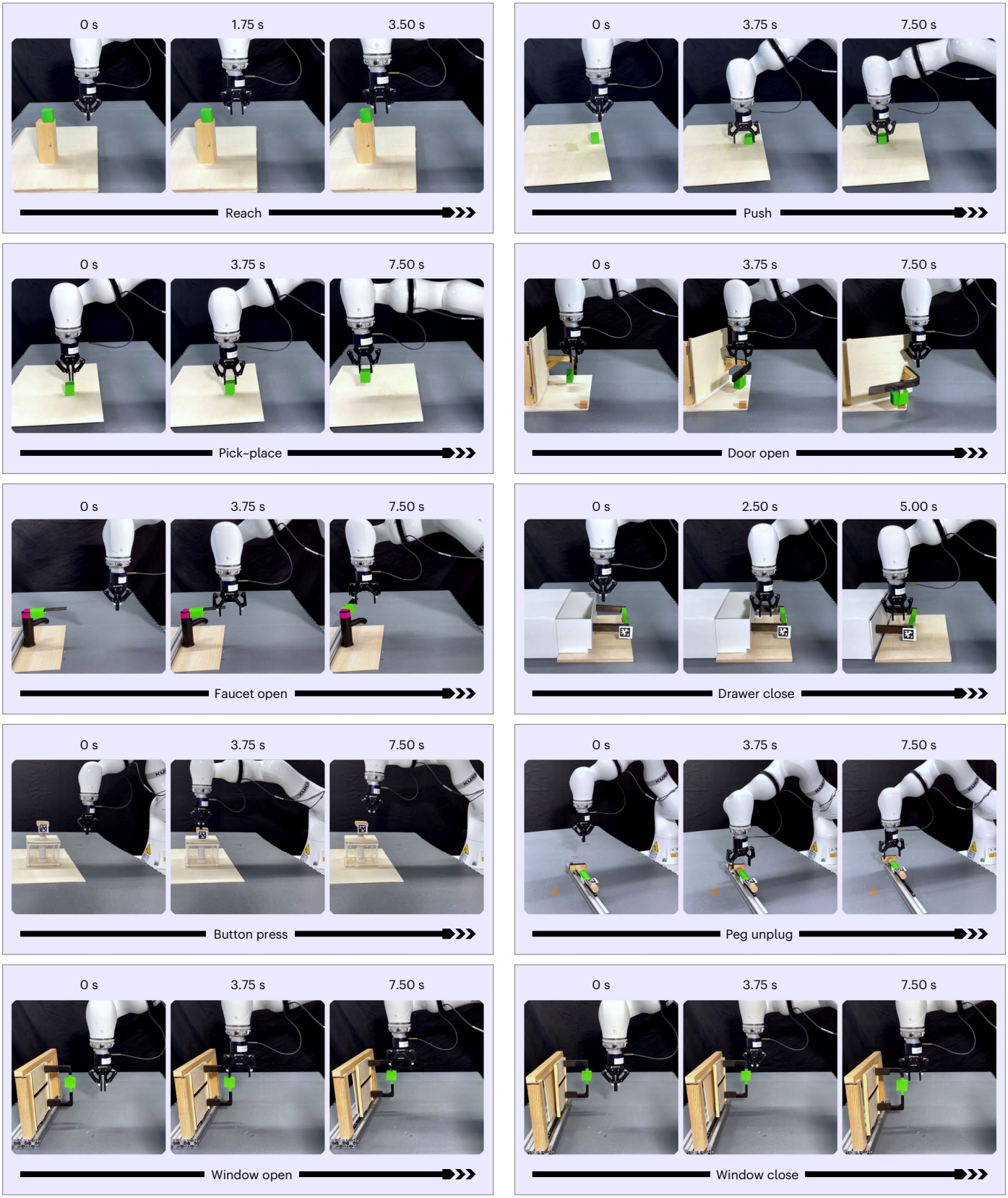


Fig. 2 | Performance on real-world single tasks. Snapshots of embodied agent on individual manipulation tasks after LRL.

in Fig. 3f. The t-SNE results demonstrate that our proposed LEGION framework can infer and identify earlier acquired knowledge and merge it into existing cluster components associated with individual tasks.

Statistics. For a quantitative assessment of our proposed framework, we present the performance results in Table 1 of each task in both conventional multi-task and lifelong training processes. The evaluation for

https://doi.org/10.1038/s42256-025-00983-2

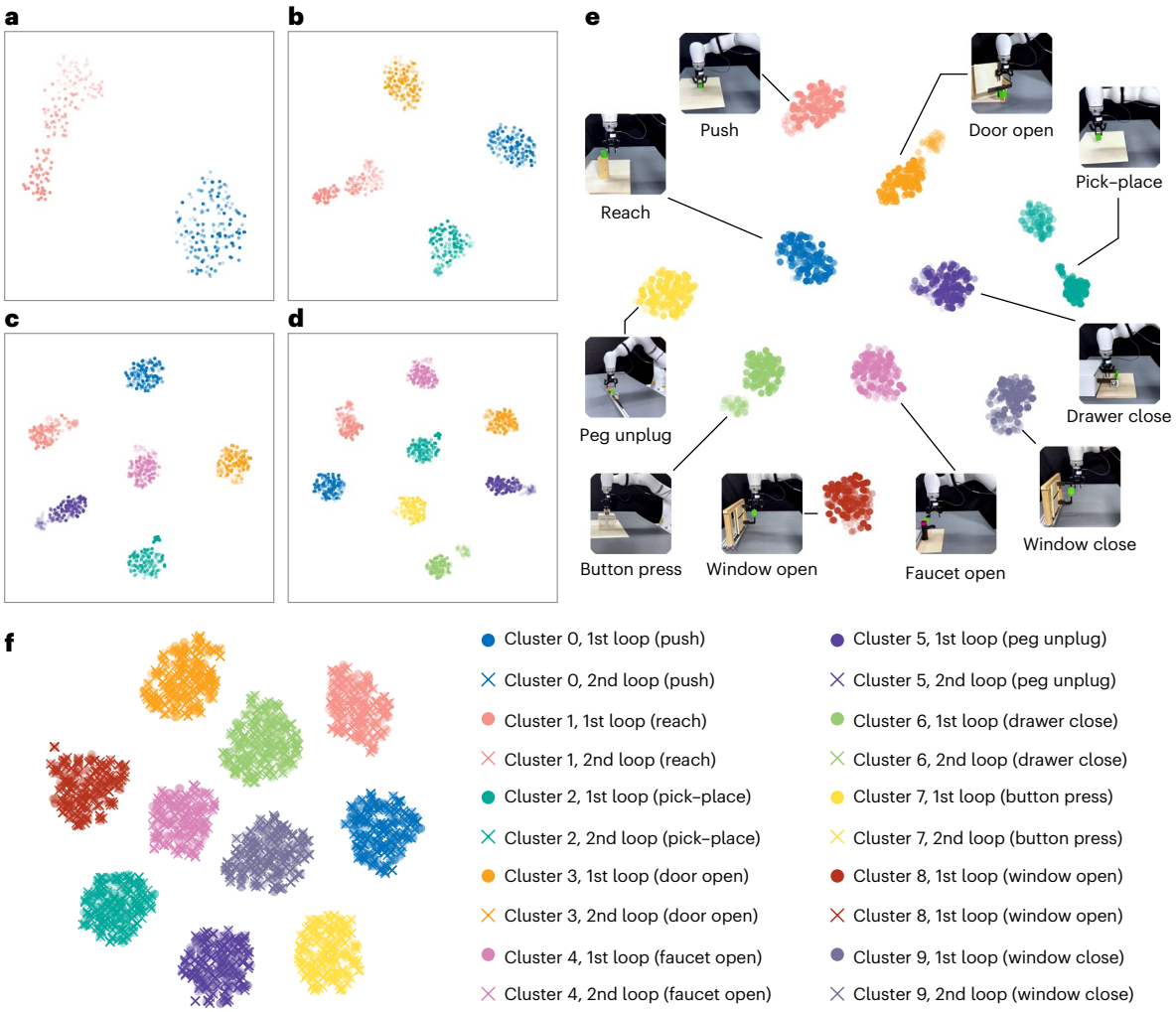


Fig. 3 | t-SNE snapshots of knowledge space. a–**e**, t-SNE projection of knowledge space after training on two tasks (**a**), four tasks (**b**), six tasks (**c**), eight tasks (**d**) and all ten tasks (**e**). **f**, t-SNE projections after the first training loop (circle) and after the second loop (cross). Notably, the inference results of the second training loop are merged into corresponding knowledge groups that are preserved during the first loop.

MTRL can lead to an in-depth understanding of our framework in terms of synchronized knowledge acquisition and preservation. In the context of lifelong learning, we utilize an easy-to-hard task ordering strategy, where the agent begins by learning fundamental tasks that serve as milestones for mastering more complex action patterns in subsequent tasks. For more details on other task ordering variations, refer to Supplementary Section 3.1. We report the success rates for each task (row-wise) after the agent trained on a one-time feeding task stream (column-wise). For example, the first column from the left side represents the agent's performance on all tasks after it has trained on the task 'reach'. Furthermore, we incorporate two additional metrics to evaluate the specific characteristics of our lifelong learnable agent, namely, 'forgetting' and 'forward transfer'. 'Forgetting' is a scalar metric in the range [−1, 1], representing how much knowledge our proposed agent may forget at the end of its lifespan. A lower value in this metric signifies better performance. 'Forward transfer', in contrast, has a range of [0, 1] that considers how much the earlier tasks knowledge aids the subsequent tasks, where a larger value indicates better performance. For more details of these metrics, refer to equations (2) and (3). We also report our agent's multi-task performance of each task listed in the right column of the table. Each datum in the table is based on trials with five random seeds. The last row calculates the average value of the data alongside column-wise. As indicated in the table, after being trained on earlier tasks, the agent maintains its performance on corresponding tasks even when trained with subsequent tasks. This implies that the acquired knowledge is effectively preserved within the model. The average success rate gradually increases, reaching 0.84. Furthermore, our proposed framework's overall average forgetting score is 0.0, showcasing its robust knowledge preservation capability. We observe that negative scores occur on tasks such as 'door open'; this is because the subsequent learning process enhances performance on previously learned tasks. For instance, after training on 'door open', the agent initially achieves a success rate of 0.4 on this task. However, after training on 'faucet open', the success rate for 'door open' improves to 0.8. This improvement is probably because the knowledge gained in understanding how to open a faucet (whether clockwise or anticlockwise) contributes positively to the door-opening task. In addition, positive forward transfer phenomena are observed in our agent's lifelong learning process. Specifically, for the task 'drawer close', earlier acquired knowledge from tasks such as 'push', 'pick–place' and 'door open' contributes to the success of 'drawer close'. For instance, the push and pull motions learned from previous tasks aid the agent in completing the drawer close task. The final average score of this metric is 0.10. In the context of a multi-task learning process, where the agent has simultaneous access to all tasks, our framework attains superior performance with a final success of 0.94 (Supplementary Sections 1 and 2).

Few-shot knowledge recall. Knowledge rehearsal is a critical component of lifelong learning. Recent studies, particularly in computer

Table 1 | Statistics of individual task success rate with easy-to-hard task ordering

Evaluation	Train after												
	Lifelong learning												Multi-task
	Reach	Push	Pick place	Door open	Faucet open	Drawer close	Button press	Peg unplug	Window open	Window close	Forgetting	Forward transfer	
Reach	1.00	1.00	0.80	0.80	0.80	1.00	1.00	1.00	1.00	1.00	0.00	NA	1.00
Push	0.00	1.00	1.00	1.00	1.00	1.00	1.00	0.80	0.80	0.80	0.20	0.00	0.80
Pick–place	0.00	0.00	0.80	1.00	0.80	0.80	1.00	0.60	1.00	0.80	0.00	0.00	0.80
Door open	0.00	0.00	0.00	0.40	0.80	0.80	0.60	0.40	0.80	0.60	−0.20	0.00	1.00
Faucet open	0.00	0.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00	1.00	0.00	0.00	1.00
Drawer close	0.00	1.00	0.80	0.80	0.00	0.60	0.80	0.80	1.00	1.00	−0.40	0.52	1.00
Button press	0.00	0.00	0.00	0.00	0.40	0.00	0.80	0.60	0.80	0.60	0.20	0.07	1.00
Peg unplug	0.00	0.00	0.00	0.00	0.00	0.20	0.00	1.00	0.60	0.60	0.40	0.03	0.80
Window open	0.00	0.00	0.40	0.00	0.60	0.00	0.20	0.00	0.80	1.00	−0.20	0.15	1.00
Window close	0.00	0.00	0.00	0.00	0.00	0.40	0.00	0.40	0.40	1.00	NA	0.13	1.00
Average	0.10	0.30	0.38	0.40	0.54	0.58	0.64	0.66	0.82	0.84	0.00	0.10	0.94

In LRL, we assess the performance of all tasks (row-wise) once the agent completes training on each one-time feeding task (column-wise). In multi-task reinforcement learning, the agent is evaluated after simultaneous training on all tasks (row-wise). Each datum is based on at least five trials, with average values reported for evaluation. The metrics 'forgetting' and 'forward transfer' are used to assess the specific characteristics of the LRL agent. 'Forgetting', in the range [−1, 1] (equation (2)), measures the extent of knowledge retention, with lower values indicating better performance. 'Forward transfer', in the range [0, 1] (equation (3)), evaluates how well earlier task knowledge supports subsequent tasks, where higher values denote better performance. NA, not available.

vision[14,16,32,33], have shown that rehearsal effectively mitigates forgetting during the learning process. However, it remains unclear whether this technique performs as well in the robotic domain, where data are continuous and time sequential. In addition, recent biological research[34–41] suggests that knowledge rehearsal aids in consolidating long-term memory and improves performance through deep memory recall, even after extended pauses (Supplementary Section 9). Building on these insights, we explore how our proposed agent performs during few-shot knowledge recall with only intermittent replay.

To demonstrate the application and potential limitations of existing replay-based lifelong learning methods in robotic reinforcement learning, we conduct comparison experiments against these baseline methods. All models use the same soft-actor–critic (SAC) policy, including the neural network backbone and shared hyperparameters. Each experiment is repeated at least five times, and the average success rate and standard deviation during evaluation are used as metrics to ensure fairness. The following baseline models are employed for comparison. (1) Reservoir. This baseline uses the 'reservoir' sampling method in the buffer to approximate the empirical distribution of observed samples. The buffer is designed to maintain a maximum data ratio of 50%. Unlike our framework, this model does not include an upstream inference and knowledge preservation module, so its policy network inputs consist of only the raw task observations without the upstream inference representations. This allows us to assess the strengths of our proposed Bayesian non-parametric knowledge space in task inference, knowledge preservation and its impact on overall task performance. (2) Perfect memory. Based on the 'reservoir' baseline, we extend the buffer size to match the total training steps, meaning that all past trajectories are stored without being forgotten or overwritten. (3) Averaged gradient episodic memory (A-GEM)[14] is a rehearsal-based method that treats lifelong learning as a constrained optimization problem. It constructs a global loss based on old training samples to ensure no loss of performance on previous tasks, projecting new sample gradients to avoid interference. Here for each base task, we maintain an episode memory of 10,000.

Figure 4a shows the average success rate during evaluation. As seen in the figure, our proposed LEGION framework consistently outperforms other methods, demonstrating a steady increase in success rate as new tasks are introduced. While perfect memory maintains a full buffer, its success rate reaches around 0.2 throughout training, showing no obvious improvement, highlighting its limitations in adapting or generalizing as the task stream progresses. Similarly, reservoir shows a flat performance curve with no notable gains, and A-GEM also underperforms in our benchmark. To further illustrate the limitations of replay-based methods in robotic LRL, Fig. 4b visualizes the data ratio in the training batch. For instance, after training on the second task 'push', the data ratio for 'push' initially remains at around 50%. However, as the agent moves through subsequent tasks, this ratio gradually decreases, eventually dropping to around 10% by the end of training. In contrast, in MTRL, the agent trains on individual tasks with a constant data ratio in the batch, ensuring stable learning conditions. This gap in the data sampling process during lifelong learning may weaken knowledge retention and lead to overall performance degradation over time. Our framework addresses this challenge by utilizing a Bayesian non-parametric knowledge inference and clustering module, which ensures consistent knowledge preservation and stable performance throughout the lifelong learning process despite fluctuating data ratios.

To assess our agent's knowledge-recall performance after pausing on tasks for a while, we selected 5 tasks from our original sequence, ordered from easy to hard, and trained the agent on them sequentially (1 million steps for each task) across 3 repeat loops: 'reach → push → faucet open → button press → window close'. In the replay buffer, we allocated space for data from only three tasks at a time. This set-up means that while training on the fourth task, data from the first task are gradually replaced by data from the fourth, and by the time the fifth task is reached, replay data from the first task are no longer available. In the second loop, we revisit the first task and compare its performance in the second loop to that in the first. This process is repeated similarly for the other tasks during the second and third loops. Extended Data Fig. 1 shows the t-SNE projections of the knowledge space after each task learning for all three loops. Figure 4c–g shows the average success rates for each task during the first loop (orange) and the subsequent second (green) and third (blue) loops. As illustrated, despite a 1 million-step pause for each task, the agent quickly re-masters them in the second and third loops, surpassing its initial performance. Our framework demonstrates faster convergence on all tasks during subsequent loops, following few-shot attempts, emphasizing the benefits of few-shot

https://doi.org/10.1038/s42256-025-00983-2

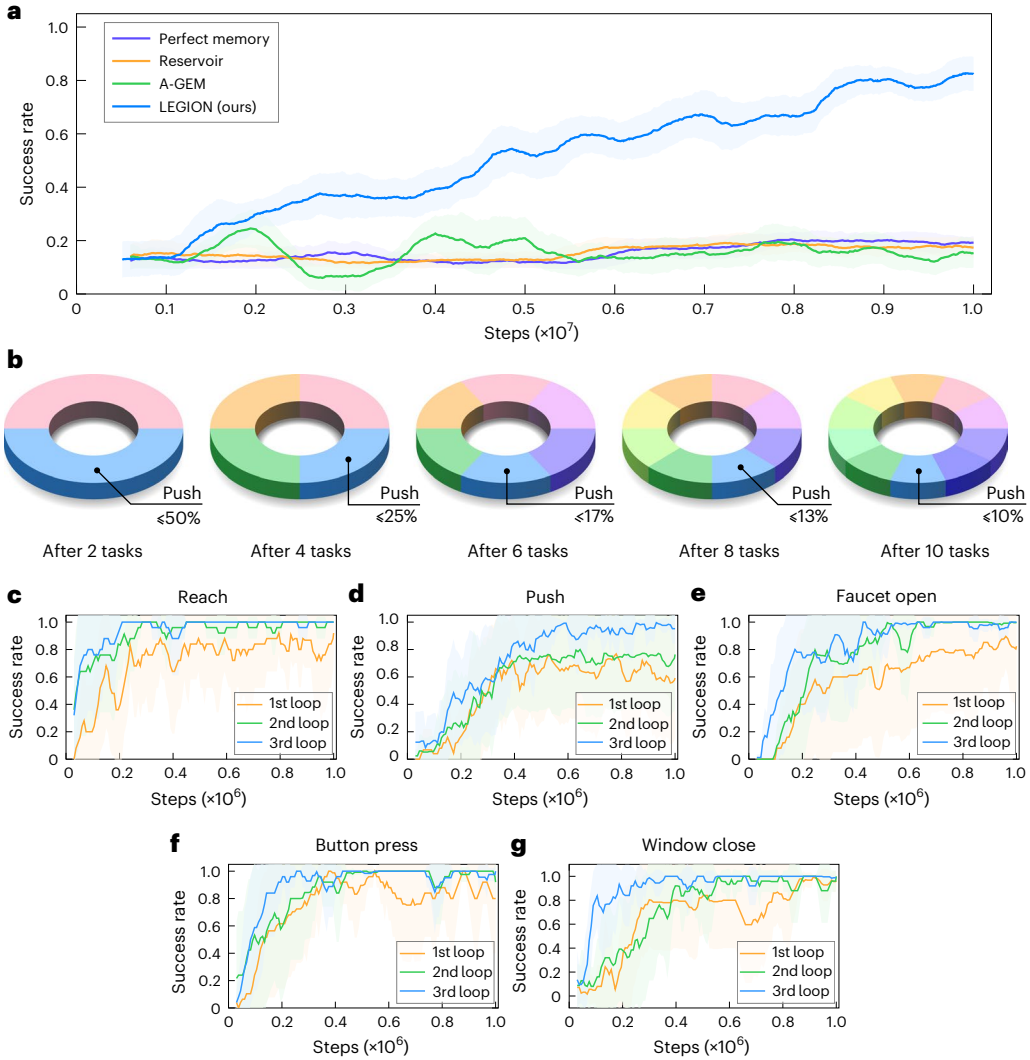


Fig. 4 | Evaluation of replay's contribution in knowledge recall. a, The average success rates from five runs of the LEGION framework compared with three replay-based lifelong learning methods: perfect memory, reservoir and A-GEM. The figure shows that LEGION consistently outperforms these methods, demonstrating a steady increase in success rate throughout the task sequence. **b**, Evolution of the 'push' task data ratio within the training batch. While the batch size remains constant, the data ratio for the 'push' task gradually decreases from an initial maximum of 50% to 10% after learning 10 tasks. **c**–**g**, Few-shot knowledge-recall performance on reach (**c**), push (**d**), faucet open (**e**), button press (**f**) and window close (**g**). The agent is trained sequentially on five selected tasks over three repeated loops, with buffer capacity limited to three tasks at a time. This configuration forces the agent to pause on each base task for 1 million steps without replay. For **a** and **c**–**g**, the data are calculated based on at least five trials, presented as mean ± standard deviation ($\mu \pm \sigma$).

memory recall. This mirrors the biological theory of mnemonics, where knowledge retention supports task re-mastery. Specifically, in the 'reach' task, despite the enforced pause, the agent consistently maintains knowledge, achieving a success rate of 0.3–0.4 at the initial evaluation checkpoint. Moreover, the agent shows an average success-rate improvement of 0.2 in the final loop compared with its initial attempts. After few-shot knowledge recalls in the third loop, the framework reaches the maximum success rate on most tasks. This improvement is attributed to effective deep memory recall enabled by our framework leveraging the DPMM.

To quantify the improvement in few-shot knowledge recall, we calculate the improvement percentage for each task (Extended Data Table 2) using equation (4). The results show that the improvement varies across tasks: 19.63% for 'reach', 6.66% for 'push', 16.77% for 'faucet open', 9.94% for 'button press' and 6.78% for 'window close' between the first and second loops. Moreover, comparing the first and third loops reveals even greater success-rate enhancements. On average, our framework shows an 11.96% improvement between the first and second loops and a substantial 21.36% improvement from the first to the third loop. These findings highlight our framework's strong capability for effective knowledge recall, rapid adaptation and improved task performance through few-shot exploration. The consistent improvement across multiple tasks underscores its robustness in re-mastering tasks and maintaining high success rates, showcasing the potential of our framework, especially the DPMM knowledge space for advancing LRL.

Discussion

Robotic lifelong learning focuses on acquiring and retaining knowledge from a continuous stream of tasks, enabling agents to progressively build more complex behaviours through knowledge integration and reuse. Our study presents a deep reinforcement learning framework that continuously accumulates knowledge from a stream of tasks, demonstrating human-like lifelong learning capability. In addition, it solves complex long-horizon tasks by combining and reapplying acquired skills, a key step towards achieving general intelligence.

In our real-world experiment with a KUKA robot arm, our agent, aided by real-time vision from a RealSense camera and language embeddings from an LLM, successfully completes a sequence of

https://doi.org/10.1038/s42256-025-00983-2

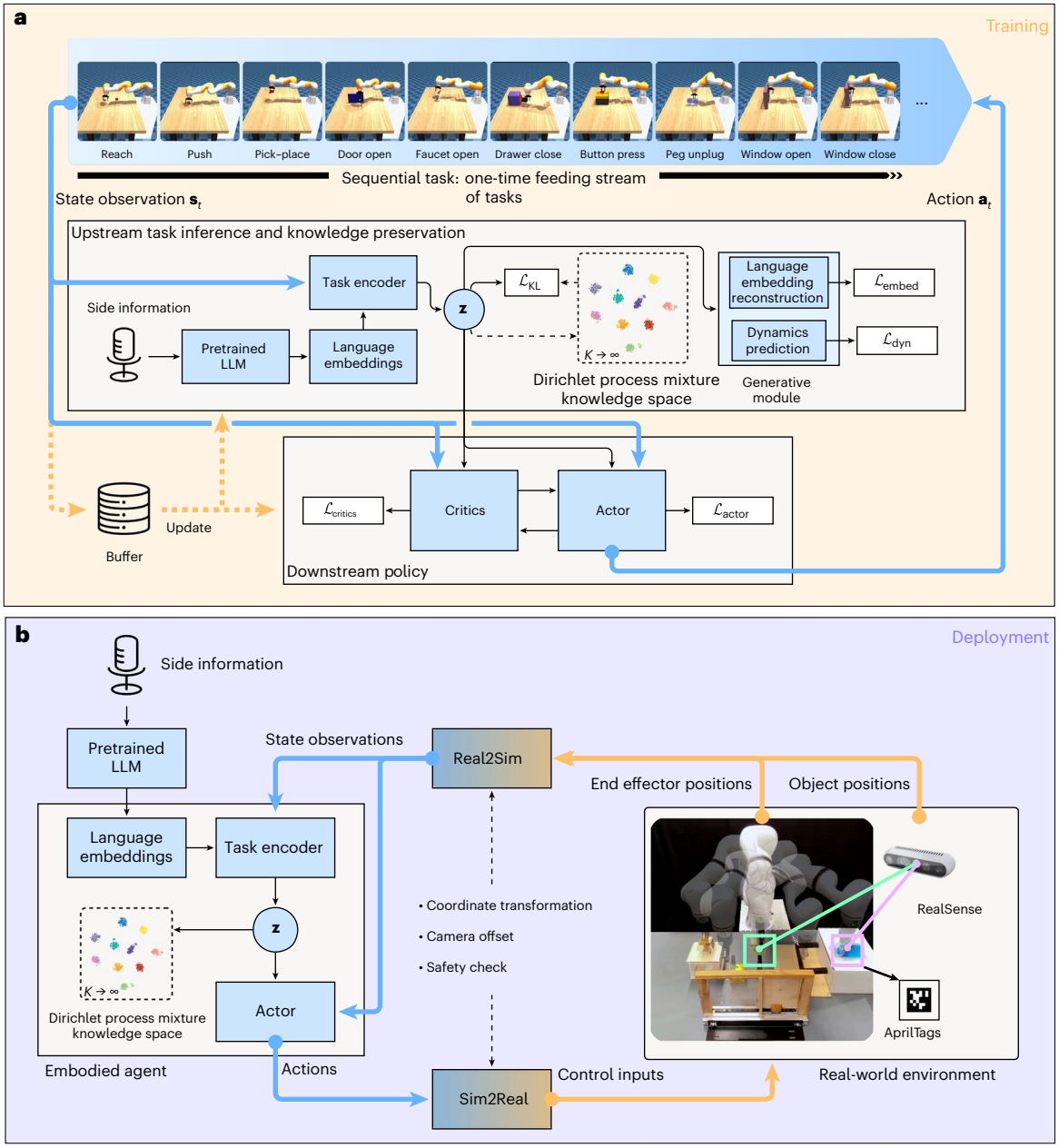


Fig. 5 | LEGION framework overview for training and deployment. a, Training. The framework receives language semantic information and environment observations as input to make policy decisions and output action patterns, it trains on only one task at a time. $\mathcal{L}$ represents the loss functions and is explained in 'Upstream task inference' in Methods. **b**, Deployment. In the real-world demonstration, the agent parameters remain frozen, the agent receives input signal from real-world hardware and outputs corresponding action signals, both 'Sim2Real' and 'Real2Sim' modules process the data to align the gap between the simulation and real world.

tasks, efficiently accumulating knowledge and demonstrating flexible, autonomous skill reapplication for long-horizon tasks without relying on predefined human demonstrations. In our LRL framework, we analyse knowledge management through both visualization and statistical perspectives. The non-parametric model in the knowledge space dynamically adjusts to new task inputs by creating or merging components, ensuring continuous knowledge preservation without prior knowledge quantity requirements. Quantitatively, the agent's success rate improves over time, demonstrating effective knowledge accumulation in LRL.

In summary, our framework LEGION (details refer to Fig. 5) excels at both preserving knowledge and inferring new tasks in its Bayesian non-parametric knowledge space during lifelong learning. Using language embeddings to aid in task inference, the agent can efficiently undertake long-horizon tasks, showcasing flexibility in addressing complex tasks based on accumulated knowledge. We acknowledge that the replay mechanism is an inherent part of our framework due to its use of SAC as policy, which relies on data sampling from a buffer and offline parameter updates. However, the replay is not strictly tied to our approach using the Bayesian non-parametric knowledge space, but rather a feature of the SAC itself. Our framework currently shows substantial improvements in few-shot exploration with intermittent replay. In the future, we plan to optimize it further to better balance stability and adaptability without relying on replay buffers, while also aiming to tackle more challenging scenarios such as zero-shot inference. Meanwhile, we acknowledge that our current framework operates in structured environments with predefined task set-ups and relies on AprilTags for perception. In future work, we aim to expand our

framework to unstructured, dynamic environments featuring diverse object arrangements and unseen objects, with the goal of enhancing the generalization and robustness of lifelong learning systems. In addition, we plan to explore applying our non-parametric knowledge space to robot learning involving multiple agents or heterogeneous embodiments (Supplementary Section 3.5), intending to achieve clustered and transferable general intelligence. As our current work assumes the reward function to be an inherent and static property of the environment, another promising future direction involves using LLMs[42–44] for continuous reward refinement during the lifelong learning process. This would enable agents to quickly adapt to entirely new control tasks. Moreover, the ability to continuously learn and preserve skills from a stream of tasks using a non-parametric knowledge space, combined with smooth and stable downstream action outputs from the diffusion model, holds potential for the development of broadly applicable large behaviour models.

Methods

Training and deployment

Training. Figure 5 a illustrates the concept overview of our proposed LEGION framework. Unlike the typical multi-task approaches, where the agent learns all tasks at once, our proposed framework can continuously gain knowledge from a stream of one-time feeding tasks. This implies that our agent can imitate the real human learning process, tackling each manipulation task one after another throughout its lifespan.

During training, we let the agent learn tasks one by one, allowing the agent to undergo 1 million training steps for each task. Importantly, we evaluate the agent's performance on all tasks every 10,000 steps by following the conventional multi-task fashion, irrespective of whether it has undergone training for these tasks or not. In our framework, we follow the off-policy training mode as it has more sampling efficiency. To achieve both preserving existing knowledge and inferring new tasks simultaneously, our proposed framework is structured hierarchically into two parts, namely, the upstream task inference and knowledge preservation module and the downstream policy learning module. The upstream module consists of the following components: the pre-trained language embedding module, the task encoder, the Dirichlet process mixture knowledge space, and the generative modules. In the simulation, we employ an offline approach where language embeddings are pre-encoded using an LLM combined with an audio recognition device and stored for training. This pre-processing step accelerates training by eliminating the need for real-time encoding, which is computationally intensive. For specific details regarding the content of the language side information, refer to Supplementary Section 7. Subsequently, the task state observations $\mathbf{s}$, which include positions of end-effector, objects and goals, are combined with the current task's language embedding $\mathbf{l}$ and sent to the task inference encoder. Following that, the generated inference results $\mathbf{z}$ are fitted by the DPMM within the knowledge space. The inferred results from the same task are clustered and stored within the same components in the DPMM, enabling knowledge preservation in our framework. When dealing with data samples from new task distributions, the DPMM can create new components to accommodate them, thereby separating them from existing clusters and supporting continual knowledge accumulation during agent lifelong learning. Simultaneously, the generative module reconstructs the language embeddings and predicts the dynamic function of the current task. This enables disentangled parameter updates between upstream and downstream modules. Moreover, an ablation study in Supplementary Section 3.4 demonstrates that the generative module plays a crucial role in stabilizing the lifelong learning process. For the downstream policy module defined in Fig. 5a, we utilize the SAC[45] as a concrete policy learning module, where the critics calculate the action value function $Q(\mathbf{s}_t, \mathbf{a}_t, \mathbf{z}_t)$ and the actor provides the corresponding action patterns $\mathbf{a}_t$ to accomplish the tasks. The inferred task results are conditioned as part of downstream inputs, contributing to more precise action pattern learning. The detailed structures of individual modules are introduced in Supplementary Section 4.

Deployment. After training in simulation environments, we implement our trained agent onto a real-world KUKA manipulator to build up an embodied lifelong learning agent. The real-world deployment overview is illustrated in Fig. 5b, where the framework includes two primary components: the embodied agent software side and the real-world hardware side. On the software side, we deploy the trained task encoder, DPMM and downstream actor to create the embodied agent. In the real-world demonstration, we utilize an online encoding approach, where human commands are processed and encoded as language embeddings to execute each task. This set-up reflects real-world usage, allowing users to issue verbal commands directly to the robot. For the hardware side, our agent's physical body comprises a KUKA iiwa with a Robotiq 2F85 gripper. In addition, we utilize a global RealSense camera at the table edge to capture object positions via AprilTags. Later, the task-related goal position is determined by the initial position of the detected object and the corresponding side information context. Python-based robot operating system controls the movement of the KUKA, with a system frequency of 20 Hz. We limit the total work steps of a single-task trajectory to 150, maintaining consistency with simulation environments. To ensure smooth communication between the software and hardware control, we employ two transformation modules, namely, 'Sim2Real' and 'Real2Sim'. These modules serve similar purposes, including safety control checks, coordinate frame transformation between simulation and the real world, hand-to-eye calibration, and camera offset set-up. A detailed experiment set-up for both simulation and deployment can be found in Supplementary Section 5. Moreover, we provide Supplementary Video 4 to introduce the implementation details of our framework for both training and deployment processes.

Language embedding. The manipulation tasks performed by a robot arm show a natural tendency towards a limited set of action patterns. On the one hand, tasks like 'push the teacup from left to right' and 'open the window in a horizontal direction' may differ in their language description, but their actual action patterns might share similar trajectories. This similarity can pose challenges during agent training, leading to inaccurate action patterns and/or misoperations in real-world performance. On the other hand, although such task-related contextual information (or side information) is often available in real-world scenarios (for example, between human communication), it is frequently overlooked in conventional reinforcement learning methods and is difficult to provide to the embodied agent without encoding of a LLM. By leveraging advancements in LLMs[3,31], our embodied LRL agent becomes more adept at utilizing this side information, like natural language-based task descriptions, to acquire generalizable skills and facilitate knowledge transfer among tasks[22]. In this study, we capture natural language side information through an external speech-recognition device. We adopt a human-in-the-loop approach to guide the embodied agent in real-world tasks. In our case, we employ one of the state-of-the-art pre-trained LLMs, RoBERTa[31], to encode the side information about manipulation tasks into language embeddings. Subsequently, these embeddings are conditioned with state observations and provided to the agent, aiding in accurate task inference and improving its execution of corresponding action patterns.

Observation space. The state observation space includes the end-effector position (three dimensions), the object pose (six dimensions) and the goal positions (three dimensions). We encode the language side information related to the task context with a pre-trained RoBERTa model, whose output has 768 dimensions.

Action space. The action space contains four dimensions, including the movement of the end-effector tip (units expressed in metres) and the open distance of the gripper. The upper and lower bounds for each dimension of action space are limited within the range [−1.00, +1.00].

Reward. Our designed scenarios involve goal-based manipulation tasks where success is determined by bringing the target object within a specified goal range. To comprehensively evaluate action patterns, the global reward for tasks is divided into distinct components. These components include rewards for reaching, grasping, pushing, pulling and pressing objects to achieve the specified targets or poses. For more details, refer to Supplementary Section 8.

Optimization. Both the upstream task inference module and downstream policy learning module are trained using Adam optimizers, and we reset the intrinsic parameters of optimizers before initiating the training on every new incoming task.

Metrics

Recent works in the conventional multi-task or meta-learning-based approaches often rely on metrics based on reward and/or success rate to evaluate agent performance. However, in the context of LRL, typical questions still remain unclear, including how much the agent may forget the previously acquired knowledge after training on subsequent tasks, or to what extent the previously acquired knowledge can aid the subsequent ones. Therefore, it becomes necessary to employ additional metrics that capture the unique characteristics of lifelong learning fashion. In our study, in addition to the success rate, we adopt two well-used metrics to access our proposed LRL framework, namely, 'forgetting' and 'forward transfer'. To further evaluate the performance of few-shot knowledge recall, we employ a normalized 'improvement' metric to quantify the agent's performance. Assuming a total of N tasks, and considering that the agent undergoes training for each task over a span of Δ steps, the cumulative global training duration across all N tasks amounts to $T = N \times \Delta$ steps.

Average success rate. In our study, as all our environments are goal-based manipulation tasks, we select the task's success rate as one of the evaluation metrics. Here, $P_i(t)$ signifies the success rate of task i at time step t. The values $P_i(\Delta(i-1))$ and $P_i(\Delta i)$ denote the success rates of task i before and after training on the same task, respectively. The overall average performance on all N tasks is calculated as follows:

$$P(t) : \frac{1}{N}\sum_{i=1}^{N} P_i(t). \tag{1}$$

Notably, P is constrained within the range [0, 1]. We also employ $P(T)$ for the final evaluation, particularly for the purpose of hyperparameter optimization and ablation study. A higher value of P corresponds to an improved performance. Moreover, we also consider the episode rewards as one of our metrics, for more details related to the rewards comparison, refer to Supplementary Sections 1 and 2.

Forgetting. This metric quantifies how much knowledge is forgotten after the agent is trained on subsequent tasks. Drawing upon recent research contributions[6,46], we introduce the forgetting (F) to assess the agent's capacity to preserve knowledge within a continuous stream of tasks. Specifically, F_i is calculated by subtracting the final success rate of task i, denoted as $P_i(T)$, from the success rate of task i after training on the task itself, represented as $P_i(\Delta i)$. The overall forgetting metric is computed as follows:

$$\begin{aligned} F_i &= \left(P_i(\Delta i) - P_i(T)\right), \\ F &= \frac{1}{N-1}\sum_{i=1}^{N-1} F_i. \end{aligned} \tag{2}$$

Notably, evaluating the forgetting of the most recently encountered task carries limited significance. Therefore, we consider only the first $N-1$ tasks for this calculation. The forgetting metric is constrained within the range $F \in [-1, 1]$. When $F > 0$ represents that the agent may have lost knowledge of prior tasks. Conversely, when $F < 0$, the backwards transfer occurs, signifying that training on subsequent tasks j (with $i < j \leqslant N$) has led to an improvement in performance on prior tasks i. For the F, a lower value indicates superior performance.

Forward transfer. Forward transfer (FT) assesses the extent to which previous tasks contribute to the learning of new ones. Inspired by recent works[1,6], we define the zero-shot forward transfer metric as follows:

$$\begin{aligned} \mathrm{FT}_i &= \frac{1}{i-1}\sum_{k=1}^{i-1} P_k(\Delta k), \\ \mathrm{FT} &= \frac{1}{N-1}\sum_{i=2}^{N} \mathrm{FT}_i. \end{aligned} \tag{3}$$

Here, the metric range is constrained to $\mathrm{FT} \in [0, 1]$, where the forward transfer for the i-th task is calculated as the average performance across tasks from $k = 1$ to $k = i-1$. It is important to emphasize that evaluating the first task holds no meaningful significance. Thus, we consider a total of $N-1$ subsequent tasks for this assessment. A higher value in this metric indicates that knowledge acquired from earlier tasks aids the agent in enhancing its performance on subsequent tasks, reflecting better performance.

Improvement of few-shot knowledge recall. To quantify the improvement statistics in few-shot knowledge recall, we calculate the improvement percentage for each task as follows. First, we computed the integral of the agent's success rate of selected two loops for each task. We then subtracted the integral from the earlier loop from the subsequent loop and normalized the difference. This normalized value, denoted as f (in the range [−1, 1]), serves as our evaluation metric for few-shot improvement. A higher value indicates better performance during the subsequent loops compared with the earlier access, whereas a lower value suggests the opposite. The specific calculation is detailed in equation (4):

$$f = \frac{1}{T \times P_{\max}}\left(\int_{t_j}^{T_j} P(t)\,\mathrm{d}t - \int_{t_i}^{T_i} P(t)\,\mathrm{d}t\right), \tag{4}$$

where $P_{\max}$ indicates the best performance value that the agent can acquire (in our case, the success rate with 1.0), $P(t)$ denotes the performance value over time, and t and T are the lower and upper bounds, respectively, of training steps with $j > i$.

Non-parametric knowledge space

In this section, we present our non-parametric knowledge space from two aspects. First, we introduce the mathematical theory behind the Dirichlet process mixture model. Following that, we introduce an online variational inference method of DPMM that is used to update the model parameters.

Dirichlet process mixtures. Bayesian non-parametric models are a class of models that allow for flexible modelling of complex data structures without making strict assumptions about the underlying distribution of the data. Unlike Bayesian parametric models (for example, Gaussian mixtures), which have a fixed number of parameters, Bayesian non-parametric models have a potentially infinite number of parameters that are determined by the data. Bayesian non-parametric models are typically based on probabilistic models that involve prior distributions over model parameters. Such prior distributions are often chosen to be flexible and allow for infinite-dimensional parameter spaces. This allows the model to adapt to the underlying structure of

the data, whether it is simple or complex. One of the typical non-parametric models is the Dirichlet process mixture model, whose parameters are determined through the Dirichlet process. The Dirichlet process is a probability distribution over probability distributions. It is used in Bayesian non-parametric to model data when the number of groups or clusters is not known a priori. Let G be a random probability measure, $\mathcal{H}$ be a base probability distribution from a parameter space Θ, and α be a positive real-valued scalar named concentration parameter. Then, G is said to be drawn from a Dirichlet process (DP) with α and $\mathcal{H}$, denoted as $G \sim \mathrm{DP}(\alpha, \mathcal{H})$. To generate the samples from a Dirichlet process, a method called the stick-breaking process is employed (Supplementary Section 10).

DPMM serves as a prominent model in the Bayesian non-parametric domain that is used to capture an infinite mixture of clusters for modelling a set of observations $\mathbf{x} = x_{1:N}$. Unlike finite mixture models in the Bayesian parametric domain, the number of components in DPMM is not predefined, but rather determined by the observations in an online fashion. In DPMM, each data x_i is sampled from a distribution $\mathcal{F}(\theta_i)$, where θ_i represents a latent variable independently drawn from a Dirichlet process prior G-based base distribution. A Dirichlet process prior introduces discreteness and clustering properties by allowing θ_i to take on repeated values. Consequently, all data points drawn with the same value of θ_i form a cluster, resulting in the natural clustering of observations. The active number of cluster components is determined by the number of unique values of θ_i, which can be dynamically inferred based on the observed data. To assign data points to clusters, each point is associated with an assignment variable υ_i. This variable takes on the value k with probability π_k, which is drawn from a categorical distribution (Cat). The generative process of DPMM can be expressed using the stick-breaking process (Supplementary Section 10.1), where the mixing proportions π can also be equivalently expressed as sampled from a generalized Ewens distribution (GEM). Specifically, the generative process of DPMM can be represented as follows:

$$\begin{aligned} \theta_k^*|\lambda &\sim \mathcal{H}(\lambda), \\ \pi|\alpha &\sim \mathrm{GEM}(\alpha), \\ \upsilon_i|\pi &\sim \mathrm{Cat}(\pi), \\ x_i|\upsilon_i &\sim \mathcal{F}(\theta_{\upsilon_i}^*). \end{aligned} \tag{5}$$

Variational inference. In this study, we focus on a variational inference-based method to estimate the true posterior of data, as they tend to offer faster and more scalable solutions compared with sampling-based methods. The fundamental concept behind variational inference is to transform the inference problem into an optimization problem. Subsequently, the aim is to uncover the underlying joint probability distribution of the unknown parameters, allowing us to explore their implicit relationships. In the case of the DPMM, as described in equation (5), the joint probability distribution of its parameters can be expressed as follows:

$$p(\mathbf{x}, \mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\beta}) = \prod_{n=1}^{N} \mathcal{F}(x_n|\theta_{\upsilon_n}) \,\mathrm{Cat}(\upsilon_n|\pi(\boldsymbol{\beta})) \prod_{k=1}^{\infty} \mathcal{B}(\beta_k|1,\alpha)\mathcal{H}(\theta_k|\lambda), \tag{7}$$

where $\mathcal{B}$ is stick-breaking process probability and β_k are corresponding random variables. As the true posterior $p(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\beta}|\mathbf{x})$ is intractable, the objective is to identify the optimal variational distribution $q^*(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\beta})$ that minimizes the Kullback–Leibler (KL) divergence from the exact conditional distribution. Instead of directly minimizing the KL divergence, we maximize the evidence lower bound (ELBO) which includes the expected log-likelihood of the data $\mathbb{E}[\log p(\mathbf{x}|\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\beta})]$ and the KL divergence between two priors $\mathbb{KL}(q(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\beta})||p(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\beta}))$. Here, we have (Supplementary Section 10.2):

$$\mathrm{ELBO}(q) = \mathbb{E}[\log p(\mathbf{x}|\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\beta})] - \mathbb{KL}(q(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\beta})||p(\mathbf{v}, \boldsymbol{\theta}, \boldsymbol{\beta})). \tag{8}$$

In the context of DPMM, grounded in the concept of variational inference, we formulate the variational distribution q under the mean-field assumption, where each latent variable possesses its variational factor, and these factors are considered independent from one another. Specifically, we have:

$$\begin{aligned} q(\mathbf{v}, \theta, \beta) &= \prod_{n=1}^{N} q(\upsilon_n|\hat{r}_n) \prod_{k=1}^{K} q(\beta_k|\hat{\alpha}_{k_1}, \hat{\alpha}_{k_0}) q(\theta_k|\hat{\lambda}_k), \\ &= \prod_{n=1}^{N} \underbrace{\mathrm{Cat}(\upsilon_n|\hat{r}_{n_1:n_K})}_{q_{\upsilon_n}} \prod_{k=1}^{K} \underbrace{\mathcal{B}(\beta_k|\hat{\alpha}_{k_1}, \hat{\alpha}_{k_0})}_{q_{\beta_k}} \underbrace{\mathcal{H}(\theta_k|\hat{\lambda}_k)}_{q_{\theta_k}}, \end{aligned} \tag{9}$$

where q_{υ_n} is a categorical factor with variational parameters $\hat{r}_{nk}$, q_{β_k} is a factor for stick-breaking proportion with parameters $\hat{\alpha}_{k_0}, \hat{\alpha}_{k_1}$, and q_{θ_k} is a base distribution factor with parameters $\hat{\lambda}_k$. In the context of variational inference, it is fundamental to recognize that the true posterior distribution is inherently infinite, and obtaining an exact representation is unfeasible, hence necessitating approximations. However, by augmenting the number of components K within the categorical factor, we can enhance the optimization of the ELBO objective, leading to a variational distribution that closely approximates the infinite posterior. To maintain computational tractability, we restrict the categorical factor to a finite set with K components ($q(\upsilon_n = k) = 0$ for $k > K$), ensuring that K is sufficiently large to encompass all potential features. Furthermore, we explore a specific scenario in which both the base distribution $\mathcal{H}$ and the cluster component distribution $\mathcal{F}$ come from the exponential family. Hughes and Sudderth[30] illustrated that in this context, it is possible to formulate the ELBO in terms of the expected mass $\hat{N}_k$ and the expected sufficient statistic $s_k(x)$ associated with each component k:

$$\begin{aligned} \mathrm{ELBO}(q) = \sum_{k=1}^{K} \Bigg[&\mathbb{E}_q[\theta_k]^{\top} s_k(x) - \hat{N}_k[a(\theta_k)] + \hat{N}_k[\log \pi_k(\beta)] - \sum_{n=1}^{N} \hat{r}_{nk} \log \hat{r}_{nk} \\ &+ \mathbb{E}_q\left[\log \frac{\mathcal{B}(\beta_k|1,\alpha)}{q(\beta_k|\hat{\alpha}_{k_1}, \hat{\alpha}_{k_0})}\right] + \mathbb{E}_q\left[\log \frac{\mathcal{H}(\theta_k|\lambda)}{q(\theta_k|\hat{\lambda}_k)}\right]\Bigg]. \end{aligned} \tag{10}$$

Subsequently, each variational factor can be iteratively updated independently. In the initial stage, we perform updates on the local variational parameters $\hat{r}_{nk}$ within q_{υ_n} for each clustering assignment. Following this step, we advance to the update of the global parameters within the stick-breaking factor q_{β_k} and the base distribution factor q_{θ_k}. We employ this coordinate ascent method to iteratively optimize the local and global parameters with the objective of maximizing the ELBO. The computation of the summary statistics $\hat{N}_k$ and $s_k(x)$ requires accessing the complete dataset. In the case of large datasets, a batch-based approach known as memoVB[30] is employed. This approach breaks down the summary statistics of the full data into a summation of the summary statistics of each batch. The non-parametric nature of the DPMM allows for flexibility in adapting to varying numbers of clusters. This characteristic enables the development of heuristics for dynamically adding or removing clusters, which proves beneficial in avoiding local optima when utilizing batch-based variational inference methods. For detailed derivations, refer to Supplementary Section 10.2.

MemoVB incorporates birth and merge moves to facilitate dynamic cluster adjustment. To create new clusters, poorly described subsamples x' from one existing cluster are collected as they pass through each batch, and a separate DPMM model with K' initial clusters is fitted. Assuming that the active number of clusters before the birth move is K, the acceptance or rejection of new cluster proposals is determined by comparing the result of assigning x' to $K + K'$ with that of assigning x' to K. In addition to the birth move, a merge move can potentially combine a pair of clusters into one. The decision to merge two clusters is based on whether the merge improves the ELBO

objective, resulting in $K-1$ clusters after the merge[30]. By integrating with this online inference method, the DPMM can consistently preserve the gained knowledge in the knowledge space from a theoretically endless stream of data, where the features or knowledge within it may steadily grow. For more details, refer to Supplementary Section 10.3.

Upstream task inference

In this section, we introduce the derivations behind our upstream task inference module (Fig. 5a). We start with the knowledge inference and preservation process of our knowledge space. Subsequently, we introduce the generative process of upstream modules that enable the disentangled and stabilized learning process.

Knowledge inference and preservation. To enable simultaneous knowledge inference and preservation in the knowledge space, we employ a DPMM + memoVB that was introduced in the previous section. The DPMM + memoVB has the advantage of being able to cluster a potentially infinite number of features based on the observations, while dynamically adapting to fit the number, shape and density of individual components. This dynamic adaptability holds great potential for preserving knowledge in a continuous stream of tasks.

Our framework employs an alternating optimization scheme to eliminate the necessity of fitting a new DPMM from scratch every time. First, we update the DPMM module using the inference result $\mathbf{z}_i$, which are sampled from the task encoder. Each update of the DPMM module takes place after certain training steps of the task encoder and generative module. Then, with the DPMM module fixed, we update the task encoder by minimizing the KL divergence, using the assigned clusters to each $\mathbf{z}_i$.

When updating the parameters of the DPMM, we perform fitting on the task inference result $\mathbf{z}_i$ obtained from the task encoder. Consider a set of state inputs $\{\boldsymbol{x}_i\}_{i=1}^{n} \in X$ with $\mathbf{x}_i = (\mathbf{s}_i, \mathbf{I}_i)$, the DPMM module in the knowledge space learns: (1) the inference results $\mathbf{z}_i$ and corresponding mapping between $\mathbf{x}_i$ and $\mathbf{z}_i$; (2) the number of K active components and their parameters $\{\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1:K}$; and (3) the cluster assignment v_i of each input, where $v_i \in \{1, \ldots, K\}$. The cluster assignments of inference results are determined jointly by the latent representation and the DPMM components. In each update, we initialize the DPMM with the parameters learned from the previous updates and apply it to new samples generated by the updated task encoder. This enables us to update the same DPMM while incorporating the latest changes in the knowledge space mappings.

During the training of the task inference module, we aim to minimize both the generative loss $\mathcal{L}_{\text{gen}}$ and the KL divergence loss $\mathcal{L}_{\text{KL}}$ in a joint manner. $\mathcal{L}_{\text{gen}}$ measures the error between the original inputs $\mathbf{x}$ and the generated samples $\mathbf{x}^*$. Meanwhile, $\mathcal{L}_{\text{KL}}$ represents the KL divergence between task encoder distribution and knowledge clustering components. To compute $\mathcal{L}_{\text{KL}}$, we first obtain the cluster assignment $v_i = k$ of each task inference results $\mathbf{z}_i$ from the current DPMM. Using the DPMM, we determine the mean and covariance of the assigned cluster k, denoted as $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ respectively. Following that, the ith inference result assigned to the component k, which is represented as $\mathbf{z}_{ik}$, is generated through the reparametric trick[47]. Notably, the hard assignment between the inference result and corresponding cluster component in knowledge space may lead to incorrect assignments for certain samples, resulting in errors when calculating the KL divergence. To address this issue, we propose the use of a soft assignment, in which we compute the probability p_{ik} of assigning the $\mathbf{z}_i$ to cluster k using the DPMM, considering all possible components $k \in \{1, 2, \ldots, K\}$. As a result, the KL divergence is defined as a weighted sum, taking into account the probabilities of each cluster assignment:

$$\mathcal{L}_{\text{KL}_i} = \sum_{k=1}^{K} p_{ik} \mathcal{L}_{\text{KL}_{ik}}, \tag{11}$$

where $\mathcal{L}_{\text{KL}_{ik}}$ indicates the KL divergence between the distribution of task encoder output and component k in DPMM, and p_{ik} represents the probability of $\mathbf{z}_i$ assigned to cluster component k. While more sophisticated weighting strategies can be employed, our empirical findings suggest that simple weighting based on probabilities is effective. For detailed derivations, refer to Supplementary Section 11.1.

Generative process. In our framework, we employ a generative module at the upstream level to facilitate a disentangled and stabilized learning process. The generative module includes two distinct components: the language embedding generation, denoted as $p_\theta(\mathbf{I}_t|\mathbf{z}_t)$, and the dynamic prediction model for corresponding manipulation task, represented as $p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{z}_t)$. Each component is interpreted by a separate multilayer perception module with general parameters $\theta \in \{\theta_{\text{embed}}, \theta_{\text{dynamics}}\}$. The detailed structures of our proposed generative modules are presented in Supplementary Figs. 18 and 19. During the training process, the language embedding decoder takes only the task inference results $\mathbf{z}_t$ as inputs and generates the language embedding tokens, donate as $\mathbf{I}^*$. These tokens are then combined with the original inputs $\mathbf{I}$ to compute the overall similarity using the sum of mean squared error as the side information loss function, denoted as $\mathcal{L}_{\text{embed}}$. Furthermore, the dynamics prediction module models the state transition function by taking the current state observation $\mathbf{s}_t$, normalized action vector $\mathbf{a}_t$, and latent variables as inputs to generate the expected state observation at the next step, denoted as $\mathbf{s}^*_{t+1}$. By comparing this prediction with the actual next observation $\mathbf{s}_{t+1}$, we obtain the loss function for dynamic prediction, which is represented by $\mathcal{L}_{\text{dyn}}$. Meanwhile, with normalization in each input dimension, we can stabilize the overall training process. Following that, the total loss of the upstream task inference module is calculated as follows:

$$\mathcal{L} = \underbrace{\zeta \mathcal{L}_{\text{dyn}} + \eta \mathcal{L}_{\text{embed}}}_{\mathcal{L}_{\text{gen}}} + \xi \mathcal{L}_{\text{KL}}. \tag{12}$$

with ζ and η representing the weighting factors for each loss function term, which regulate the importance of each generative module. ξ represents the disentangled factor of the KL divergence term. In summary, the generation module facilitates a disentangled learning process unaffected by downstream policy training. This set-up stabilizes the exploration process of the downstream policy module, particularly during the initial steps of each task where noise is inevitable. Simultaneously, regenerating language embeddings and modelling the state transition function contribute to the agent learning more accurate action patterns for individual manipulation tasks. For detailed derivations of the loss functions related to the generative process, refer to Supplementary Section 11.2.

Reporting summary

Further information on research design is available in the Nature Portfolio Reporting Summary linked to this article.

Data availability

All data needed to evaluate the conclusions are present in the article and the Supplementary Information. Microsoft Excel (Version 2405, Build 17628.20110, 64-bit) was utilized to analyse, interpret and summarize the statistical results. The source data used to present this work are available on Zenodo at https://doi.org/10.5281/zenodo.14265089 (ref. 48) or via GitHub at https://github.com/Ghiara/LEGION.

Code availability

The code used for training and evaluation, which supports the conclusions of this study, is publicly available via Zenodo at https://doi.org/10.5281/zenodo.14265089 (ref. 48) or via GitHub at https://github.com/Ghiara/LEGION.

References

1. Khetarpal, K., Riemer, M., Rish, I. & Precup, D. Towards continual reinforcement learning: a review and perspectives. *J. Artif. Intell. Res.* **75**, 1401–1476 (2022).
2. Croitoru, F.-A., Hondru, V., Ionescu, R. T. & Shah, M. Diffusion models in vision: a survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **45**, 10850–10869 (2023).
3. Achiam, J. et al. GPT-4 technical report. Preprint at https://arxiv.org/abs/2303.08774 (2023).
4. Zhao, J. et al. Autonomous driving system: a comprehensive survey. *Expert Syst. Appl.* **242**, 122836 (2024).
5. Aljundi, R., Babiloni, F., Elhoseiny, M., Rohrbach, M. & Tuytelaars, T. Memory aware synapses: learning what (not) to forget. In *Proc. European Conference on Computer Vision (ECCV)* 139–154 (2018).
6. Chaudhry, A., Dokania, P. K., Ajanthan, T. & Torr, P. H. Riemannian walk for incremental learning: understanding forgetting and intransigence. In *Proc. European Conference on Computer Vision (ECCV)* 532–547 (2018).
7. Chen, Z. & Liu, B. *Lifelong Machine Learning* Vol. 1 (Springer Nature, 2022).
8. Delange, M. et al. A continual learning survey: sefying forgetting in classification tasks. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* 1–1 (IEEE, 2021).
9. Kirkpatrick, J. et al. Overcoming catastrophic forgetting in neural networks. *Proc. Natl Acad. Sci. USA* **114**, 3521–3526 (2017).
10. Nguyen, C. V., Li, Y., Bui, T. D. & Turner, R. E. Variational continual learning. In *Proc. 6th International Conference on Learning Representations* (OpenReview.net, 2018).
11. Mallya, A. & Lazebnik, S. PackNet: adding multiple tasks to a single network by iterative pruning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 7765–7773 (IEEE, 2018).
12. Fernando, C. et al. PathNet: evolution channels gradient descent in super neural networks. Preprint at https://arxiv.org/abs/1701.08734 (2017).
13. Rosenfeld, A. & Tsotsos, J. K. Incremental learning through deep adaptation. *IEEE Trans. Pattern Anal. Mach. Intell.* **42**, 651–663 (2018).
14. Chaudhry, A., Ranzato, M., Rohrbach, M. & Elhoseiny, M. Efficient lifelong learning with A-GEM. In *Proc. 7th International Conference on Learning Representations* (OpenReview.net, 2019).
15. Lopez-Paz, D. & Ranzato, M. Gradient episodic memory for continual learning. In *Advances in neural information processing systems* (eds Guyon, I. et al.) Vol. 30 (Curran Associates, Inc., 2017).
16. Rebuffi, S.-A., Kolesnikov, A., Sperl, G. & Lampert, C. H. iCaRL: incremental classifier and representation learning. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition* 2001–2010 (IEEE, 2017).
17. Parisi, G. I., Kemker, R., Part, J. L., Kanan, C. & Wermter, S. Continual lifelong learning with neural networks: a review. *Neural Netw.* **113**, 54–71 (2019).
18. Yu, T. et al. Meta-world: a benchmark and evaluation for multi-task and meta reinforcement learning. In *Proc. Conference on Robot Learning* 1094–1100 (PMLR, 2020).
19. Vithayathil Varghese, N. & Mahmoud, Q. H. A survey of multi-task deep reinforcement learning. *Electronics* **9**, 1363 (2020).
20. Yu, T. et al. Gradient surgery for multi-task learning. *Adv. Neural Inf. Process. Syst.* **33**, 5824–5836 (2020).
21. Yang, R., Xu, H., Wu, Y. & Wang, X. Multi-task reinforcement learning with soft modularization. *Adv. Neural Inf. Process. Syst.* **33**, 4767–4777 (2020).
22. Sodhani, S., Zhang, A. & Pineau, J. Multi-task reinforcement learning with context-based representations. In *Proc. 38th International Conference on Machine Learning* 9767–9779 (PMLR, 2021).
23. Perez, E., Strub, F., De Vries, H., Dumoulin, V. & Courville, A. FiLM: visual reasoning with a general conditioning layer. In *Proc. AAAI Conference on Artificial Intelligence* 32 (AAAI, 2018).
24. Borsa, D., Graepel, T. & Shawe-Taylor, J. Learning shared representations in multi-task reinforcement learning. Preprint at https://arxiv.org/abs/1603.02041 (2016).
25. Bing, Z., Lerch, D., Huang, K. & Knoll, A. Meta-reinforcement learning in non-stationary and dynamic environments. *IEEE Trans. Pattern Anal. Mach. Intell.* **45**, 3476–3491 (2022).
26. Rakelly, K., Zhou, A., Finn, C., Levine, S. & Quillen, D. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *Proc. 36th International Conference on Machine Learning, Proc. Machine Learning Research* Vol. 97 (eds Chaudhuri, K. & Salakhutdinov, R.) 5331–5340 (PMLR, 2019).
27. Finn, C., Abbeel, P. & Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning* 1126–1135 (PMLR, 2017).
28. Melo, L. C. Transformers are meta-reinforcement learners In *Proc. 39th International Conference on Machine Learning, Proc. Machine Learning Research* Vol. 162 (eds Chaudhuriet, K. et al.) 15340–15359 (PMLR, 2022).
29. Nam, T., Sun, S.-H., Pertsch, K., Hwang, S. J. & Lim, J. J. Skill-based meta-reinforcement learning. In *Proc. 10th International Conference on Learning Representations* (OpenReview.net, 2022).
30. Hughes, M. C. & Sudderth, E. Memoized online variational inference for dirichlet process mixture models. *Adv. Neural Inf. Process. Syst.* **26**, (2013).
31. Liu, Y. RoBERTa: a robustly optimized bert pretraining approach. Preprint at https://arxiv.org/abs/1907.11692 (2019).
32. Chaudhry, A., Gordo, A., Dokania, P., Torr, P. & Lopez-Paz, D. Using hindsight to anchor past knowledge in continual learning. *Proc. AAAI Conf. Artif. Intell.* **35**, 6993–7001 (2021).
33. Rolnick, D., Ahuja, A., Schwarz, J., Lillicrap, T. & Wayne, G. Experience replay for continual learning. In *Proc. 33rd International Conference on Neural Information Processing Systems* 350–360 (Curran Associates Inc., 2019).
34. Kudithipudi, D. et al. Biological underpinnings for lifelong learning machines. *Nat. Mach. Intell.* **4**, 196–210 (2022).
35. Hadsell, R., Rao, D., Rusu, A. A. & Pascanu, R. Embracing change: continual learning in deep neural networks. *Trends Cogn. Sci.* **24**, 1028–1040 (2020).
36. Wilson, M. A. & McNaughton, B. L. Reactivation of hippocampal ensemble memories during sleep. *Science* **265**, 676–679 (1994).
37. Ji, D. & Wilson, M. Coordinated memory replay in the visual cortex and hippocampus during sleep. *Nat. Neurosci.* **10**, 100–107 (2007).
38. Rasch, B. & Born, J. Maintaining memories by reactivation. *Curr. Opin. Neurobiol.* **17**, 698–703 (2007).
39. Lee, J. L. Memory reconsolidation mediates the strengthening of memories by additional learning. *Nat. Neurosci.* **11**, 1264–1266 (2008).
40. Jacoby, L. L. & Bartz, W. H. Rehearsal and transfer to LTM. *J. Verbal Learning Verbal Behav.* **11**, 561–565 (1972).
41. Dark, V. J. & Loftus, G. R. The role of rehearsal in long-term memory performance. *J. Verbal Learning Verbal Behav.* **15**, 479–490 (1976).
42. Ma, Y. J. et al. Eureka: human-level reward design via coding large language models. In *Proc. 12th International Conference on Learning Representations* (OpenReview.net, 2024).
43. Ma, Y. J. et al. DrEureka: language model guided sim-to-real transfer. In *Robotics: Science and Systems (RSS)* (2024).
44. Yu, W. et al. Language to rewards for robotic skill synthesis. In *Proc. 7th Conference on Robot Learning* (eds Tan, J. et al) 374–404 (PMLR, 2023).
45. Haarnoja, T. et al. Soft actor–critic algorithms and applications. Preprint at https://arxiv.org/abs/1812.05905 (2018).

46. Wołczyk, M., Zając, M., Pascanu, R., Kuciński, Ł. & Miłoś, P. Continual world: a robotic benchmark for continual reinforcement learning. *Adv. Neural Inf. Process. Syst.* **34**, 28496–28510 (2021).
47. Higgins, I. et al. Beta-VAE: learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations* (Poster) **3**, (2017).
48. Meng, Y. et al. LEGION. *Zenodo* https://doi.org/10.5281/zenodo.14265088 (2024).

Acknowledgements

This study was supported in part by the financial support from the National Natural Science Foundation of China under grant number U22B2042 (to F.S.), and the Bavarian State Ministry for Economic Affairs, Regional Development and Energy (StMWi) for the Lighthouse Initiative KI.FABRIK (Phase 1: Infrastructure as well as the research and development programme under grant number DIK0249, to A.K.).

Author contributions

Y.M., Z.B., X.Y., K.C., K.H., Y.G., F.S. and A.K. developed the framework and performed all experiments. Y.M., Z.B. and X.Y. completed the manipulator deployment. Y.M., Z.B., X.Y., K.C., K.H., Y.G., F.S. and A.K. wrote the paper together. Y.M., X.Y. and K.C. prepared real-world models. K.H., F.S. and A.K. provided funding.

FundingInformation

Open access funding provided by Technische Universität München.

Competing interests

The authors declare no competing interests.

Additional information

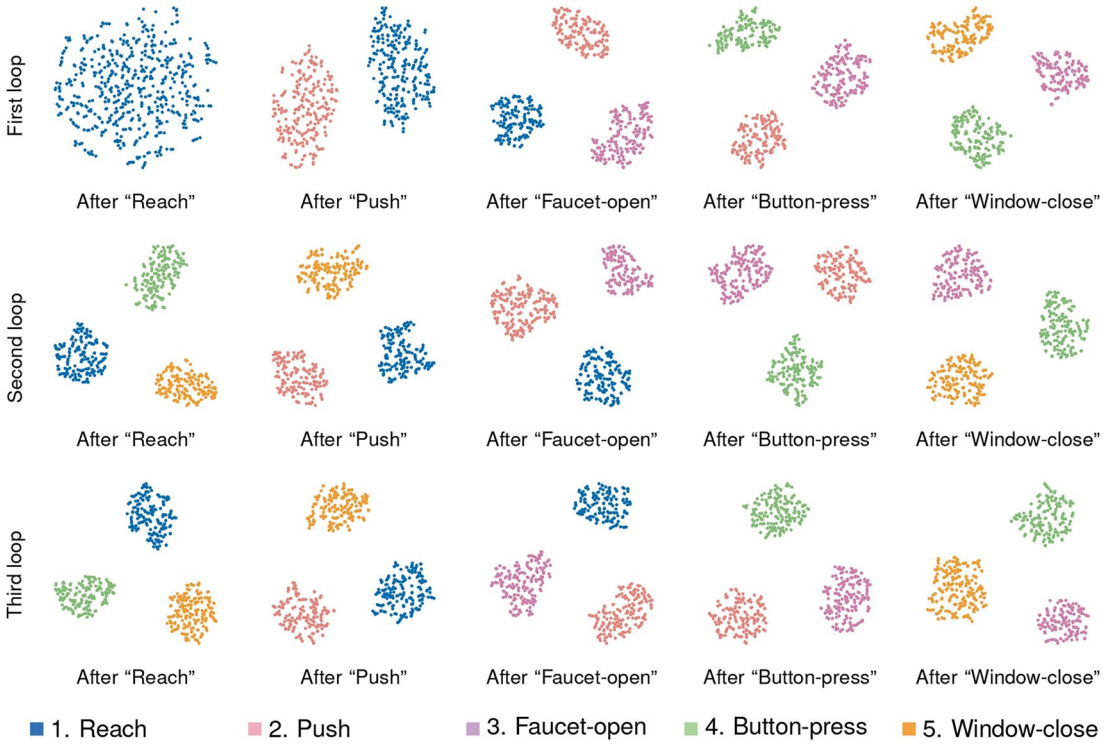
Extended data is available for this paper at https://doi.org/10.1038/s42256-025-00983-2.

Supplementary information The online version contains supplementary material available at https://doi.org/10.1038/s42256-025-00983-2.

Correspondence and requests for materials should be addressed to Zhenshan Bing, Kai Huang, Yang Gao or Fuchun Sun.

Peer review information *Nature Machine Intelligence* thanks Guangliang Li, and the other, anonymous, reviewer(s) for their contribution to the peer review of this work.

https://doi.org/10.1038/s42256-025-00983-2



Extended Data Fig. 1 | t-SNE projections of buffer data inference results after each base task training. The data is randomly sampled from the buffer and fed into the task encoder to do the inference. In the buffer we reserve a place for only three tasks, the new incoming inputs will overwrite the earliest data in the buffer. We use this method to force the agent to pause on the corresponding task for a period of time and evaluate its few-shot performance in the subsequent loops (few-shot revisit and knowledge recall). The DPMM dynamically adjusts its knowledge clustering components using the 'birth' and 'merge' heuristics, fitting model parameters based on observed data. This approach eliminates the need to predetermine or set any assumptions about the number of tasks the agent may encounter.

https://doi.org/10.1038/s42256-025-00983-2

Extended Data Table 1 | The average success rate for each proposed manipulation task in the real-world setup

Individual real-world manipulation task average success rate (3 trials per task)									
Reach	Push	Pick-place	Door-open	Faucet-open	Drawer-close	Button-press	Peg-unplug	Window-open	Window-close
1.00	0.67	0.67	0.67	1.00	1.00	1.00	0.67	1.00	1.00

Each data point is calculated based on three real-world trials, and we report the average value for comparison.

https://doi.org/10.1038/s42256-025-00983-2

Extended Data Table 2 | Improvement percentage (%) from the few-shot evaluation

Individual task few-shot improvement (%)			
Loop-to-loop comparison	**1st → 2nd**	**2nd → 3rd**	**1st → 3rd**
Reach	19.63 %	3.08 %	22.71 %
Push	6.66 %	15.03 %	21.69 %
Faucet-open	16.77 %	7.58 %	24.34 %
Button-press	9.94 %	4.07 %	14.01 %
Window-close	6.78 %	17.30 %	24.07 %
Average	+11.96 %	+9.41 %	+21.36 %

The values are computed using equation (4) from the main text.

nature portfolio

Corresponding author(s): Zhenshan Bing, Kai Huang, Yang Gao, Fuchun Sun

Last updated by author(s): Dec 6, 2024

Reporting Summary

Nature Portfolio wishes to improve the reproducibility of the work that we publish. This form provides structure for consistency and transparency in reporting. For further information on Nature Portfolio policies, see our Editorial Policies and the Editorial Policy Checklist.

Statistics

For all statistical analyses, confirm that the following items are present in the figure legend, table legend, main text, or Methods section.

n/a	Confirmed	
☐	☒	The exact sample size (*n*) for each experimental group/condition, given as a discrete number and unit of measurement
☒	☐	A statement on whether measurements were taken from distinct samples or whether the same sample was measured repeatedly
☒	☐	The statistical test(s) used AND whether they are one- or two-sided *Only common tests should be described solely by name; describe more complex techniques in the Methods section.*
☒	☐	A description of all covariates tested
☒	☐	A description of any assumptions or corrections, such as tests of normality and adjustment for multiple comparisons
☐	☒	A full description of the statistical parameters including central tendency (e.g. means) or other basic estimates (e.g. regression coefficient) AND variation (e.g. standard deviation) or associated estimates of uncertainty (e.g. confidence intervals)
☐	☒	For null hypothesis testing, the test statistic (e.g. *F*, *t*, *r*) with confidence intervals, effect sizes, degrees of freedom and *P* value noted *Give P values as exact values whenever suitable.*
☒	☐	For Bayesian analysis, information on the choice of priors and Markov chain Monte Carlo settings
☒	☐	For hierarchical and complex designs, identification of the appropriate level for tests and full reporting of outcomes
☒	☐	Estimates of effect sizes (e.g. Cohen's *d*, Pearson's *r*), indicating how they were calculated

Our web collection on statistics for biologists contains articles on many of the points above.

Software and code

Policy information about availability of computer code

Data collection	We use widely adopt Mujoco simulator (Version 2.0.0) based robot learning benchmark "Meta-World" (https://github.com/Ghiara/Metaworld-KUKA-IIWA-R800) for training our proposed framework in simulation environments. We modified the original code to fit to our lifelong learning requirements. State-of-the-art baseline learning algorithms are used in this work, including reinforcement learning frameworks for multi-task, Soft-Actor-Critic (SAC), Film, CARE. For all multitask RL baseline models, we use the open source code repository "MTRL" (https://github.com/facebookresearch/mtrl) with main branch version. For lifelong learning domain, we leverage and modify the code of algorithms for Elastic Weight Consolidation (EWC), Averaged Gradient Episodic Memory (A-GEM) , PackNet and L2. For all lifelong learning baseline models, we use open source benchmark "Continual World" (https://github.com/awarelab/continual_world) with main branch version for comparison. For real-world deployment, we use ROS1 to control and collect the trajectories of KUKA iiwa R820 robot arm. No third party software was used for data collection itself. We provide all data (.csv files) stemming from the data acquisition from experiments, which were subsequently used in statistical analysis and evaluation. Our framework open source code repository as well as source data presented in the paper can be visited at: https://github.com/Ghiara/LEGION.
Data analysis	Microsoft Excel Version 2405 Build 17628.20110 64-bit was used to interpret, read and summaries the statistical results.

For manuscripts utilizing custom algorithms or software that are central to the research but not yet described in published literature, software must be made available to editors and reviewers. We strongly encourage code deposition in a community repository (e.g. GitHub). See the Nature Portfolio guidelines for submitting code & software for further information.

Data

Policy information about availability of data

All manuscripts must include a data availability statement. This statement should provide the following information, where applicable:

- Accession codes, unique identifiers, or web links for publicly available datasets
- A description of any restrictions on data availability
- For clinical datasets or third party data, please ensure that the statement adheres to our policy

All data needed to evaluate the conclusions are present in the paper and the Supplementary Materials. Microsoft Excel (Version 2405, Build 17628.20110, 64-bit) was utilized to analyze, interpret, and summarize the statistical results. The source data used to present this work are available on Zenodo at https://doi.org/10.5281/zenodo.14265089 or GitHub at https://github.com/Ghiara/LEGION/tree/master/data

Human research participants

Policy information about studies involving human research participants and Sex and Gender in Research.

Reporting on sex and gender	Not Applicable
Population characteristics	Not Applicable
Recruitment	Not Applicable
Ethics oversight	Not Applicable

Note that full information on the approval of the study protocol must also be provided in the manuscript.

Field-specific reporting

Please select the one below that is the best fit for your research. If you are not sure, read the appropriate sections before making your selection.

☒ Life sciences ☐ Behavioural & social sciences ☐ Ecological, evolutionary & environmental sciences

For a reference copy of the document with all sections, see nature.com/documents/nr-reporting-summary-flat.pdf

Life sciences study design

All studies must disclose on these points even when the disclosure is negative.

Sample size	For each trial in both lifelong learning reinforcement and multitask reinforcement learning, we let agent run for each manipulation task 1 Million steps (steps=1,000,000), respectively. This setup follows the configurations from the baseline model repositories to ensure a fair comparison across all models. For knowledge space evaluation using t-SNE, each colored cluster group represents a complete motion trajectory for each nonparametric task (with sample N=150 for each task), which was clustered to the corresponding Gaussian group. The choice of N=150 is based on our experimental setup, where the complete motion trajectory for a robot operation task consists of 150 steps. To evaluate the differences in the low-dimensional mapping of the robot's entire trajectory across different tasks, we selected the maximum number of samples (i.e., 150) for each task.
Data exclusions	No data were excluded from our study. All data are presented in the manuscript and supplementary materials.
Replication	To ensure the reproducibility of our proposed framework. We execute repeat trials for each experiment, in which we maintain same hyper-parameter setup including learning rate, batch-size, model size and backbone. We run for each trial with different random seeds sampled from (0,1,2,3,4). Each experiment is repeat at least five times. To ensure fair baseline comparison, we fine-tuned each baseline framework in our benchmark, the trials for baselines used same learning rate and random seeds as our framework. The results can be reproduced by the code we provided in github: https://github.com/Ghiara/LEGION.
Randomization	All models trained and subsequently evaluated were randomized.
Blinding	No datasets were excluded from our study. All datasets/task environments are used to evaluate the performance of framworks/baseline models, no blinding tests are applied in the study.

Reporting for specific materials, systems and methods

We require information from authors about some types of materials, experimental systems and methods used in many studies. Here, indicate whether each material, system or method listed is relevant to your study. If you are not sure if a list item applies to your research, read the appropriate section before selecting a response.

Materials & experimental systems

n/a	Involved in the study
☒	☐ Antibodies
☒	☐ Eukaryotic cell lines
☒	☐ Palaeontology and archaeology
☒	☐ Animals and other organisms
☒	☐ Clinical data
☒	☐ Dual use research of concern

Methods

n/a	Involved in the study
☒	☐ ChIP-seq
☒	☐ Flow cytometry
☒	☐ MRI-based neuroimaging