LIBERO: Benchmarking Knowledge Transfer in Lifelong Robot Learning

Bo Liu*[1], Yifeng Zhu*[1], Chongkai Gao*[2], Yihao Feng[3], Qiang Liu[1], Yuke Zhu[1], Peter Stone[1,4]

***Abstract*—Lifelong learning offers a promising paradigm of building a generalist agent that learns and adapts over its lifespan. Unlike conventional lifelong problems in image and text domains, where distribution shifts between tasks mainly exist in the form of declarative knowledge about entities and concepts, lifelong learning in sequential decision-making (LLDM) also requires the transfer of procedural knowledge (i.e., knowledge about actions and behaviors). To advance research in LLDM, we introduce LIBERO, a benchmark of lifelong learning for robot manipulation. Specifically, LIBERO highlights five key research topics centered around LLDM: 1) how to efficiently transfer declarative knowledge, procedural knowledge, or the mixture of both; 2) how to design effective policy architectures and 3) effective algorithms for continual learning over an evergrowing number of tasks; 4) the robustness of a lifelong learner with respect to task ordering; and 5) the effects of model pretraining in learning subsequent tasks. We develop a procedural generation pipeline to create four task suites (130 tasks in total), which we use to investigate these topics. To support sample-efficient learning, we also provide high-quality human-teleoperated demonstration data for all tasks. Our extensive experiments uncovered several *unexpected* discoveries: the sequential fine-tuning method outperforms existing lifelong learning methods in forward transfer, visual encoders excel at specific types of knowledge transfer, and naive supervised pretraining can hinder downstream LLDM performance.**[1]

***Index Terms*—lifelong learning, robot learning**

I. Introduction

Lifelong learning (LL) is a practical way of training a generalist agent that can master a diverse set of tasks over its lifetime. The main body of the LL literature has examined the transfer of *declarative* knowledge about entities and concepts [1], [2]. However, how agents transfer both *declarative* and *procedural* knowledge, which pertains to the ability to perform tasks in policy learning for decision-making tasks, is not well-understood. For instance, when a robot fails to grab juice from the fridge, a task it has learned before, it can be caused by either forgetting what the juice looks like (declarative knowledge), or by forgetting how to open the fridge or grasp the juice (procedural knowledge).

This paper addresses this research gap by investigating lifelong learning in decision-making (LLDM), i.e., how to train agents that continually learn and adapt to an evergrowing number of sequential decision-making tasks. Specifically, we study the *robot manipulation* domain, as manipulation tasks require a robot agent not only to understand various concepts through perception but also to reason about how its actions influence the environment.

To enable a systematic study of knowledge transfer in LLDM, we introduce LIfelong learning BEchmark on RObot manipulation tasks (LIBERO). LIBERO is designed with a procedural generation pipeline to create simulated robot manipulation tasks, where the taxonomy of tasks is mined from human egocentric activity videos [3]. The pipeline leverages a PDDL-based scene description language [4] that can programmatically generate an infinite number of tasks. Human activity-inspired tasks intrinsically encompass entangled knowledge transfer. For systematic evaluations, we create 130 tasks grouped into four suites in LIBERO. The first three task suites (LIBERO-SPATIAL, LIBERO-OBJECT, and LIBERO-GOAL), with 30 tasks in total, examine distribution shifts in one of three aspects: type of objects, spatial arrangement of objects, and the task goals, respectively. To solve these tasks, the agent must continually acquire and transfer knowledge for recognizing objects, reasoning about the spatial relationship among objects, and executing a task. The remaining 100 tasks (LIBERO-100) involve distribution shifts on all three aspects and are hence more challenging for knowledge transfer. Natural language descriptions are used as the task identifier as they can easily specify open-ended tasks. Figure 1 illustrates all four task suites. To support efficient learning, we provide high-quality, human-teleoperated demonstration data for all 130 tasks. Compared to existing benchmarks for meta/multitask learning, LIBERO is explicitly designed for studying LLDM in robot manipulation and is more scalable and extendable, with the flexibility to create an infinite number of tasks through automated task generation (See Section A for more detailed comparisons of LIBERO with prior work).

Based on LIBERO, we investigate five major research topics regarding LLDM (also illustrated in the bottom row of Figure 1): **1)** knowledge transfer in the presence of different types of distribution shift; **2)** neural architecture design; **3)** lifelong learning algorithm design; **4)** robustness of the learner to task ordering; and **5)** how to leverage pre-trained models in LLDM (See Figure 1). We perform extensive experiments across different policy architectures and different lifelong learning algorithms. Among the **surprising observations** arising from our experiments are:

1) The design of policy architecture is equally critical to that of lifelong learning algorithms. Transformers are better at abstracting temporal information than recurrent neural networks. Vision transformers work well on tasks

Identify applicable funding agency here. If none, delete this.

[1]Bo Liu, Yifeng Zhu, Qiang Liu, Yuke Zhu, and Peter Stone are with the Department of Computer Science, the University of Texas at Austin. [2]Chongkai Gao is with Tsinghua University. [3]Yihao Feng is with Salesforce AI. [4]Peter Stone is also with Sony AI. * indicates equal contributions.

[1]Check the website at https://sites.google.com/view/liberobenchmark/home.

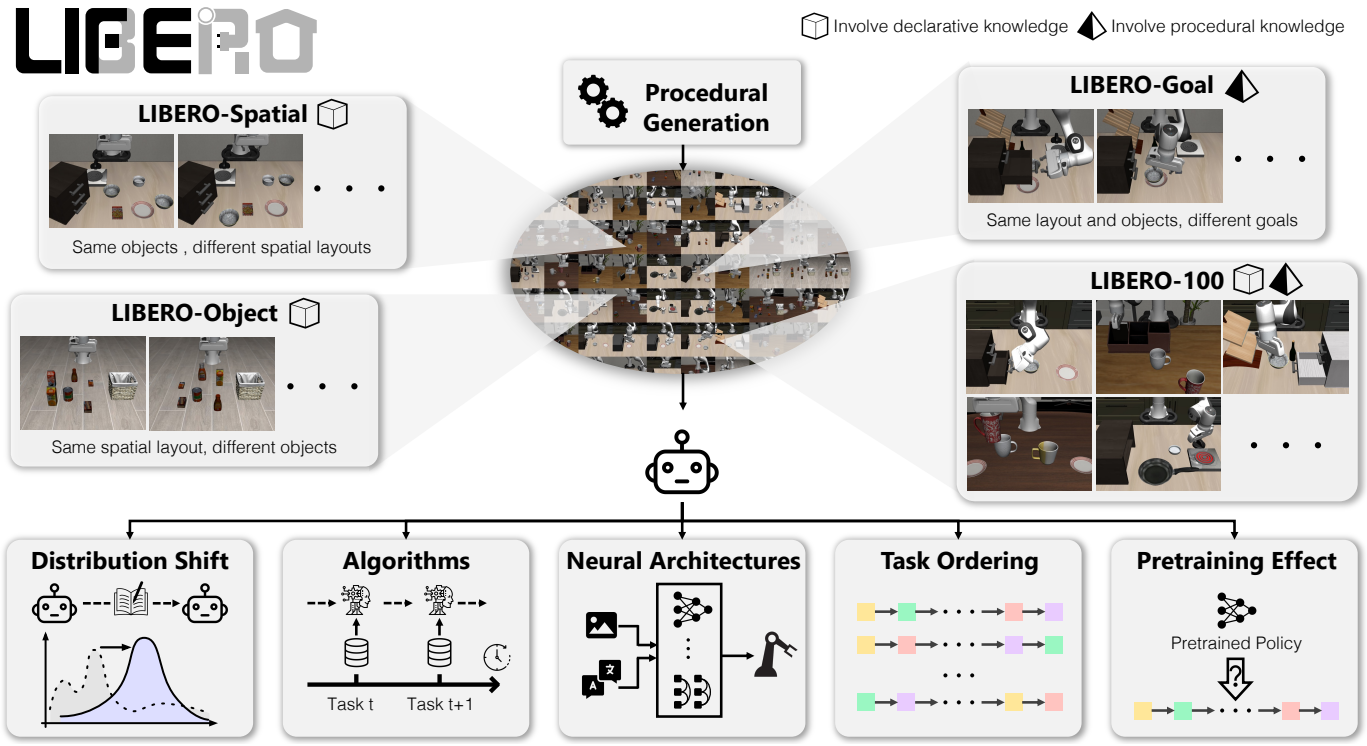


Fig. 1. LIBERO includes four procedurally-generated task suites (**top**): LIBERO-SPATIAL, LIBERO-OBJECT, LIBERO-GOAL, LIBERO-100. The first three task suites induce distribution shifts along specific axes. For instance, tasks in LIBERO-SPATIAL have the same types of objects involved and the same underlying task goal but require the agent to reason about different spatial locations of the objects. LIBERO-100 induces a distribution shift along all three axes. With these four suites, we systematically investigate the five key research topics in LLDM (**bottom**).

with rich visual information (e.g., a variety of objects). Convolution networks work well when tasks mainly require the use of procedural knowledge.

2) While the lifelong learning algorithms we evaluated are effective at preventing catastrophic forgetting, they may not perform on par with naive sequential fine-tuning in forward transfer.
3) Experiments show no significant benefits of using three pretrained language embeddings over a task embedding that lacks semantic information about the task.
4) The most basic supervised pretraining on a rich offline dataset can have a *negative* impact on the learner's downstream performance in LLDM.

II. RESEARCH TOPICS IN LLDM

In this section, we outline five major research topics in LLDM that motivate the design of the LIBERO task suites and our benchmarking study.

a) (T1) *Knowledge Transfer under Different Types of Distribution Shift:* LLDM requires transferring both declarative and procedural knowledge in decision-making tasks. For example, in the task "put the ketchup next to the plate in the basket," the robot must understand the concepts of "ketchup," the location of the plate and basket, and how to physically "put" the ketchup in the basket. The first two entail *declarative* knowledge, while the last entails *procedural* knowledge. However, when the agent learns this task, all these different types of knowledge are intertwined, making it difficult to determine the cause of failure, whether it is a lack of understanding of the concept "ketchup" (Object), an inability to localize the ketchup (Spatial), or an inability to execute the "put" motion (Goal). In Section III-B, we design three task suites that allow us to investigate knowledge transfer under the distribution shift of spatial understanding, object categories, and task goals in a disentangled manner.

b) (T2) *Neural Architecture Design:* It has been established that the design of neural network architectures plays a critical role in learning in both partially observable [5] and lifelong settings [6]. In this study, we consider a multi-modal input for the robot, including both a task indicator in natural language and observations in raw images. As a result, designing a policy network that can 1) extract *spatial* information about objects at the current time step, 2) combine *temporal* information across multiple steps within a task, and 3) transfer only relevant knowledge when learning new tasks, is a significant yet largely open research challenge.

c) (T3) *Lifelong Learning Algorithm Design:* Similar to lifelong learning in vision and language problems, given a fixed policy network, algorithm design is also an important problem in LLDM. In addition, because error can compound over time steps during policy execution in MDPs (due to their sequential nature), it is harder for the agent to retain knowledge from previously learned tasks, compared with in

a standard supervised learning setting like continual image classification [7]. As such, we consider the design of lifelong learning algorithms to be an open area of research in LLDM.

d) (T4) *Robustness to Task Ordering:* It is known that a good curriculum [8] can result in more efficient multi-task policy learning [9]. But in reality, we cannot assume the robot always receives tasks in the order most beneficial for efficient learning. Therefore, enabling the robot's lifelong learning performance to be robust to any task ordering is critical for applying such agents in the real world.

e) (T5) *Usage of Pretrained Models:* In reality, it is reasonable to pretrain a robot in factories before deployment [10]. However, it is not well-understood whether/how pretraining could benefit subsequent lifelong learning in LLDM.

III. LIBERO

This section first introduces the procedural generation pipeline in LIBERO that allows the never-ending creation of tasks to facilitate lifelong learning research. Then we describe the four task suites generated from this pipeline that benchmark both disentangled and entangled knowledge transfer in LLDM. Finally, we list the five algorithms and three neural architectures we use to systematically study how the algorithmic and the architectural designs impact the lifelong learning performance regarding the five research topics mentioned in Section II.

A. Procedural Generation of Tasks

To facilitate research in lifelong learning settings where an agent continually learns and adapts to new tasks, a procedural generation pipeline is necessary that allows the creation of an enormous number of tasks. However, it is non-trivial to systematically scale up the number of tasks while keeping created tasks relevant. LIBERO includes a procedural generation pipeline to use human activities to generate tasks in the simulator. Human activities naturally provide a large set of tasks that share common semantic concepts and motion behaviors. This pipeline harnesses the activity data through three modules that: 1) extract behavioral templates from the language annotations from the activity datasets, and then generate language instructions based on the object availability in the simulator; 2) specify an initial state distribution of objects and 3) specify task goals using a propositional formula that align with the language instructions. We use a modular robot manipulation simulator, `Robosuite` [11], that allows easy integration of our generation pipeline. Figure 2 illustrates an example of this pipeline, and each component is expanded upon below.

a) Behavioral Templates and Instruction Generation: Human activities serve as a fertile source of tasks that can inspire and generate a vast number of manipulation tasks. Both human activities and manipulation tasks require perceiving visual observations and interacting with objects on a sophisticated level, making activities a natural fit for the source of task creation in robot manipulation. However, a direct mapping from human activities to simulated manipulation tasks is not feasible due to the differences in human and robot embodiment and object model availability in the simulator. As a workaround, we choose a large-scale activity dataset, Ego4D [3], which includes a large variety of everyday activities with language annotations. We pre-process the dataset by extracting the language descriptions and then summarize them into a large set of commonly used language templates. After this pre-processing step, we use the templates and select objects available in the simulator to generate a set of language instructions. For example, we can generate "Open the drawer of the cabinet" from the template "Open ...".

b) Initial State Distribution (μ_0): Once a set of language instructions is generated, we create layouts of the scene so that the instructions are feasible in a given scene configuration. We first select the scenes that match the objects and behaviors mentioned in the language instructions. For example, a kitchen scene is selected for a language instruction *Open the top drawer of the cabinet and put the bowl in it*. The initial state distribution μ_0 is specified in a scene-description language based on PDDL [4], [12] which includes the object instances, initial placement distributions, and the scene layout (See Figure 2-**(A)**). A list of predicates describing the initial states of the objects is specified so the simulator can load in the information in PDDL format and instantiates task instances by sampling object configurations from the initial state distributions (See Figure 2-**(B)**).

c) Goal Specifications (g): Based on the μ_0 and the language instruction, we specify a goal for each instruction using a conjunction of predicates. Every predicate can be in two forms: 1) unary predicates that describe the properties of an object, such as `Open` or `Turnoff`, or 2) binary predicates that describe spatial relations between objects, such as `On` or `In`. Figure 2-**(C)** illustrates how the PDDL scene description file specifies the goal through the conjunction of predicates, and the simulator terminates when all predicates are true.

B. Task Suites

While procedural generation supports the generation of an unlimited number of tasks, we offer a fixed set of task suites because it enables apple-to-apple comparison for any future work regarding the research topics in Section II. Given this fixed set of tasks, we provide teleoperated expert demonstrations, which will not be available for tasks beyond the provided suites. The four task suites in LIBERO are LIBERO-SPATIAL, LIBERO-OBJECT, LIBERO-GOAL, and LIBERO-100. The first three task suites are well-curated to disentangle the transfer of *declarative* and *procedural* knowledge (as mentioned in (T1)), while LIBERO-100 is a suite of 100 tasks with entangled knowledge transfer. Note that the procedural generation pipeline can scale up the number of tasks in the future with ease.

a) LIBERO-X*:* LIBERO includes three task suites, each with 10 tasks [2] that are designed to evaluate different aspects of distribution shift: LIBERO-SPATIAL (shift

[2]We use 10 tasks because that is enough to observe drastic forgetting, but is few enough to maintain computation efficiency.

in spatial arrangements), LIBERO-OBJECT (shift in object types), and LIBERO-GOAL (shift in task goals). The first two suites evaluate the *declarative* knowledge transfer through varying spatial arrangements or object categories, while the third suite evaluates the *procedural* knowledge transfer through varying task goals. Specifically, LIBERO-SPATIAL evaluates if a policy can associate visual features and language with knowledge of spatial understanding by using consistent object layouts except for two bowls with unique arrangements in each task. The goal for the robot is to identify one of the bowls based on its spatial location or relation to other objects and place it on the plate. LIBERO-OBJECT evaluates the ability of a policy to associate visual features and language instruction with object types. The tasks are to pick-place unique objects from a list of objects with different categories. Tasks in LIBERO-GOAL have the same layouts but different goals, which keep the involved *declarative* knowledge constant and entail different behaviors. Such designs allow evaluation of a policy's ability to transfer *procedural* knowledge. More details are in Appendix C.

b) LIBERO-100*:* LIBERO-100 contains 100 tasks that entail diverse object interactions and versatile motor skills. In this paper, we split LIBERO-100 into 90 short-horizon tasks and 10 long-horizon tasks, and we refer to the latter one as LIBERO-LONG. We apply lifelong learning algorithms to LIBERO-LONG in all studies while the 90 short-horizon tasks serve as the data source for pre-training policies to study the effect of pre-training models (T5).

C. Lifelong Learning Algorithms

To evaluate the efficacy of different algorithmic designs to retain knowledge in LLDM, we study three representative lifelong learning algorithms, Experience Replay (ER) [13], Elastic Weight Consolidation (EWC) [7], and PACKNET [14]. ER is a memory-based method to store important past data for new task learning. EWC is a regularization-based method that constrains the neural network update. PACKNET is a method based on a dynamic architecture that updates the neural network architectures on the fly across different tasks. We additionally implement two baseline methods, sequential learning (SEQL) and multitask learning (MTL), which serve as a lower bound and upper bound for lifelong learning algorithms, respectively.

D. Neural Network Architectures

We examine three neural network architectures for policies in LIBERO. These architectures, namely RESNET-RNN, RESNET-T, and VIT-T, all follow a similar design that integrates visual, temporal, and linguistic information. All the linguistic information, the language instructions, are encoded into language embeddings using Bert [15]. The RESNET-RNN architecture is based on LSTMs and has been shown to be effective in imitation learning for vision-based manipulation [5]. It uses a ResNet as the visual backbone that encodes per-step visual observations and an LSTM as the temporal backbone to process a sequence of encoded visual information. The language instruction is incorporated into the ResNet features using the FiLM method [16] and added to the LSTM inputs, respectively. RESNET-T architecture [17] uses a similar ResNet-based visual backbone, but a transformer decoder [18] as the temporal backbone to process outputs from ResNet, which are a temporal sequence of visual tokens. The language embedding is treated as a separate token in inputs to the transformer alongside the visual tokens. The VIT-T architecture [19], which is widely used in visual-language tasks, uses a Vision Transformer (ViT) as the visual backbone and a transformer decoder as the temporal backbone. The language embedding is treated as a separate token in inputs of both ViT and the transformer decoder. All the temporal backbones output a latent vector for every decision-making step. We compute the multi-modal distribution over manipulation actions using a Gaussian-Mixture-Model (GMM) based output head [5], [20]. In the end, a robot executes a policy by sampling a continuous value for end-effector action from the output distribution. Figure 3 visualizes the three architectures.

For all the lifelong learning algorithms and neural architectures, we use behavioral cloning (BC) [21] to train policies for individual tasks (See (**??**)). BC allows for efficient policy learning such that we can study lifelong learning algorithms without excessive computational resources. To train BC, we provide 50 trajectories of high-quality demonstrations for every single task in the generated task suites. The demonstrations are collected by human experts through teleoperation with 3Dconnexion Spacemouse.

IV. EXPERIMENT OVERVIEW

Experiments are conducted as an initial study for the five research topics mentioned in Section II. Specifically, we focus on addressing the following research questions:

- **Q1**: How do different neural policies/lifelong learning algorithms perform under specific distribution shifts?
- **Q2**: To what extent does neural architecture impact knowledge transfer in LLDM, and are there any discernible patterns in the specialized capabilities of each architecture?
- **Q3**: How do existing algorithms from the literature of lifelong supervised learning perform on LLDM tasks?
- **Q4**: To what extent does language embedding affect knowledge transfer in LLDM?
- **Q5**: How do different lifelong learning algorithms compare in terms of robustness to task ordering in LLDM?
- **Q6**: Can supervised pretraining improve downstream lifelong learning performance in LLDM?

The detailed results/findings are in Appendix D.

V. CONCLUSIONS

This paper introduces LIBERO, a new benchmark in robot manipulation domain for LLDM research. LIBERO introduces 130 tasks in 4 task suites that support studies on different types of knowledge transfer. We conduct a comprehensive set of experiments on policy and algorithm designs across 5 study topics which sheds light on future research in LLDM.

REFERENCES

[1] Z. Mai, R. Li, J. Jeong, D. Quispe, H. Kim, and S. Sanner, "Online continual learning in image classification: An empirical survey," *Neurocomputing*, vol. 469, pp. 28–51, 2022.

[2] M. Biesialska, K. Biesialska, and M. R. Costa-Jussa, "Continual lifelong learning in natural language processing: A survey," *arXiv preprint arXiv:2012.09823*, 2020.

[3] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu *et al.*, "Ego4d: Around the world in 3,000 hours of egocentric video," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 18 995–19 012.

[4] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. E. Vainio, Z. Lian, C. Gokmen, S. Buch, K. Liu *et al.*, "Behavior: Benchmark for everyday household activities in virtual, interactive, and ecological environments," in *Conference on Robot Learning*. PMLR, 2022, pp. 477–490.

[5] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[6] S. I. Mirzadeh, A. Chaudhry, D. Yin, T. Nguyen, R. Pascanu, D. Gorur, and M. Farajtabar, "Architecture matters in continual learning," *arXiv preprint arXiv:2202.00275*, 2022.

[7] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[8] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, "Curriculum learning for reinforcement learning domains: A framework and survey," *arXiv preprint arXiv:2003.04960*, 2020.

[9] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009, pp. 41–48.

[10] L. P. Kaelbling, "The foundation of efficient robot learning," *Science*, vol. 369, no. 6506, pp. 915–916, 2020.

[11] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, "robosuite: A modular simulation framework and benchmark for robot learning," *arXiv preprint arXiv:2009.12293*, 2020.

[12] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "Pddl-the planning domain definition language," 1998.

[13] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P. K. Dokania, P. H. Torr, and M. Ranzato, "On tiny episodic memories in continual learning," *arXiv preprint arXiv:1902.10486*, 2019.

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

[15] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[16] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville, "Film: Visual reasoning with a general conditioning layer," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[17] Y. Zhu, A. Joshi, P. Stone, and Y. Zhu, "Viola: Imitation learning for vision-based manipulation with object proposal priors," *arXiv preprint arXiv:2210.11339*, 2022.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[19] W. Kim, B. Son, and I. Kim, "Vilt: Vision-and-language transformer without convolution or region supervision," in *International Conference on Machine Learning*. PMLR, 2021, pp. 5583–5594.

[20] C. M. Bishop, "Mixture density networks," 1994.

[21] M. Bain and C. Sammut, "A framework for behavioural cloning." in *Machine Intelligence 15*, 1995, pp. 103–129.

[22] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[23] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[25] V. Lomonaco and D. Maltoni, "Core50: a new dataset and benchmark for continuous object recognition," in *Conference on Robot Learning*. PMLR, 2017, pp. 17–26.

[26] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, "Glue: A multi-task benchmark and analysis platform for natural language understanding," *arXiv preprint arXiv:1804.07461*, 2018.

[27] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4938–4947.

[28] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[29] O. E. L. Team, A. Stooke, A. Mahajan, C. Barros, C. Deck, J. Bauer, J. Sygnowski, M. Trebacz, M. Jaderberg, M. Mathieu *et al.*, "Open-ended learning leads to generally capable agents," *arXiv preprint arXiv:2107.12808*, 2021.

[30] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski, "Vizdoom: A doom-based ai research platform for visual reinforcement learning," in *2016 IEEE conference on computational intelligence and games (CIG)*. IEEE, 2016, pp. 1–8.

[31] M. Wołczyk, M. Zając, R. Pascanu, Ł. Kuciński, and P. Miłoś, "Continual world: A robotic benchmark for continual reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 28 496–28 510, 2021.

[32] S. Powers, E. Xing, E. Kolve, R. Mottaghi, and A. Gupta, "Cora: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents," *arXiv preprint arXiv:2110.10067*, 2021.

[33] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," in *International conference on machine learning*. PMLR, 2020, pp. 2048–2056.

[34] M. Samvelyan, R. Kirk, V. Kurin, J. Parker-Holder, M. Jiang, E. Hambro, F. Petroni, H. Küttler, E. Grefenstette, and T. Rocktäschel, "Minihack the planet: A sandbox for open-ended reinforcement learning research," *arXiv preprint arXiv:2109.13202*, 2021.

[35] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 740–10 749.

[36] M. Wołczyk, M. Zając, R. Pascanu, Ł. Kuciński, and P. Miłoś, "Disentangling transfer in continual reinforcement learning," *arXiv preprint arXiv:2209.13900*, 2022.

[37] B. Ermis, G. Zappella, M. Wistuba, and C. Archambeau, "Memory efficient continual learning with transformers," 2022.

[38] S. James, Z. Ma, D. R. Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3019–3026, 2020.

[39] T. Mu, Z. Ling, F. Xiang, D. Yang, X. Li, S. Tao, Z. Huang, Z. Jia, and H. Su, "Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations," *arXiv preprint arXiv:2107.14483*, 2021.

[40] J. A. Mendez, M. Hussing, M. Gummadi, and E. Eaton, "Composuite: A compositional reinforcement learning benchmark," *arXiv preprint arXiv:2207.04136*, 2022.

[41] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[42] J. Yoon, E. Yang, J. Lee, and S. J. Hwang, "Lifelong learning with dynamically expandable networks," *arXiv preprint arXiv:1708.01547*, 2017.

[43] C.-Y. Hung, C.-H. Tu, C.-E. Wu, C.-H. Chen, Y.-M. Chan, and C.-S. Chen, "Compacting, picking and growing for unforgetting continual learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[44] L. Wu, B. Liu, P. Stone, and Q. Liu, "Firefly neural architecture descent: a general approach for growing neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 22 373–22 383, 2020.

[45] A. Chaudhry, P. K. Dokania, T. Ajanthan, and P. H. Torr, "Riemannian walk for incremental learning: Understanding forgetting and intransigence," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 532–547.

[46] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, "Progress & compress: A scalable framework for continual learning," in *International Conference on Machine Learning*. PMLR, 2018, pp. 4528–4537.

[47] D. Lopez-Paz and M. Ranzato, "Gradient episodic memory for continual learning," *Advances in neural information processing systems*, vol. 30, 2017.

[48] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny, "Efficient lifelong learning with a-gem," *arXiv preprint arXiv:1812.00420*, 2018.

[49] P. Buzzega, M. Boschini, A. Porrello, D. Abati, and S. Calderara, "Dark experience for general continual learning: a strong, simple baseline," *Advances in neural information processing systems*, vol. 33, pp. 15 920–15 930, 2020.

[50] M. De Lange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 44, no. 7, pp. 3366–3385, 2021.

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

[52] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[53] N. Díaz-Rodríguez, V. Lomonaco, D. Filliat, and D. Maltoni, "Don't forget, there is more than forgetting: new metrics for continual learning," *arXiv preprint arXiv:1810.13166*, 2018.

[54] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[55] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[56] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 627–635.

[57] S. Greydanus, A. Koul, J. Dodge, and A. Fern, "Visualizing and understanding atari agents," in *International conference on machine learning*. PMLR, 2018, pp. 1792–1801.

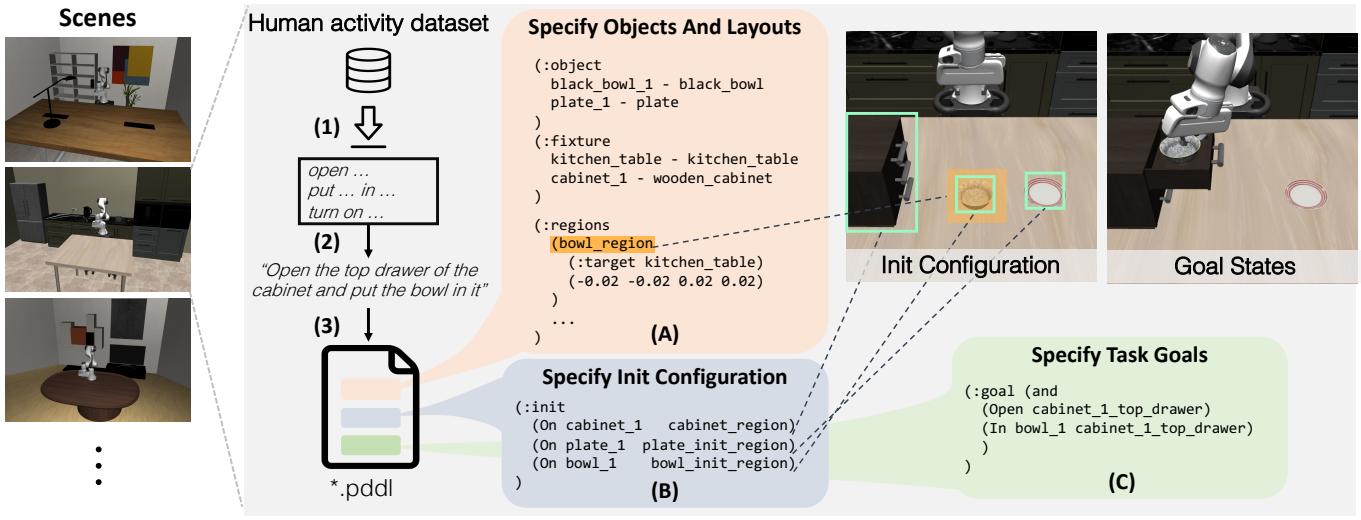


Fig. 2. This figure highlights the essential components of the procedural generation pipeline in LIBERO. **(1)** We extract behavioral templates from a large-scale human activity dataset, Ego4D; **(2)** We generate language instructions by selecting the objects whose models are available in the simulators; **(3)** We select a scene (shown on the left) that is appropriate for the language instruction, and programmatically generate a PDDL-based scene description file that specifies: **(A)** the initial configuration distribution μ_0 which includes object categories, placement regions; **(B)** the initial states of objects in the format of a list of predicates; and **(C)** the goal in a logic proposition format that consists of predicates. The goal is satisfied when all the predicates are true. The upper right part of the figure shows screenshots of initial configurations and the configuration when the goal is satisfied.

APPENDIX A
RELATED WORK

This section provides an overview of existing benchmarks for lifelong learning and robot learning. We refer the reader to Appendix B-B for a detailed review of lifelong learning algorithms.

a) Lifelong Learning Benchmarks: Pioneering work has adapted standard vision or language datasets for studying LL. This line of work includes image classification datasets like MNIST [22], CIFAR [23], and ImageNet [24]; segmentation datasets like Core50 [25]; and natural language understanding datasets like GLUE [26] and SuperGLUE [27]. Besides supervised learning datasets, video game benchmarks (e.g., Atari [28], XLand [29], and VisDoom [30]) in reinforcement learning (RL) have also been used for studying LL. However, LL in standard supervised learning does not involve procedural knowledge transfer, while RL problems in games do not represent human activities. ContinualWorld [31] modifies the 50 manipulation tasks in MetaWorld for LL. CORA [32] builds four lifelong RL benchmarks based on Atari, Procgen [33], MiniHack [34], and ALFRED [35]. Prior works have also analyzed different components in a LL agent [6], [36], [37], but they do not focus on robot manipulation problems.

b) Robot Learning Benchmarks: A variety of robot learning benchmarks have been proposed to address challenges in meta learning (MetaWorld, yu2020meta), causality learning (CausalWorld, ahmed2020causalworld), multi-task learning [38], policy generalization to unseen objects [39], and compositional learning [40]. Compared to existing benchmarks in lifelong learning and robot learning, LIBERO is uniquely designed to address the research topics of LLDM through a carefully curated set of task suites. The benchmark includes a large number of tasks based on everyday human activities that feature rich interactive behaviors with a diverse range of objects. Additionally, the tasks in LIBERO are procedurally generated, making the benchmark scalable and adaptable. Moreover, the provided high-quality human demonstration dataset in LIBERO supports and encourages learning efficiency.

APPENDIX B
IMPLEMENTED NEURAL ARCHITECTURES AND LIFELONG LEARNING ALGORITHMS

A. Neural Architectures

In Section III-D, we outlined the neural network architectures utilized in our experiments, namely RESNET-RNN, RESNET-T, and VIT-T. The specifics of each architecture are illustrated in Figure 3. Furthermore, Table II, III, and IV display the hyperparameters for the architectures used throughout all of our experiments.

Neural Policy Arch.	ResNet-RNN ResNet-T ViT-T
Lifelong Learning Algo.	SeqL EWC [7] ER [13] PackNet [14] MTL

TABLE I
THE IMPLEMENTED NEURAL POLICY ARCHITECTURES AND THE LIFELONG LEARNING ALGORITHMS IN LIBERO.

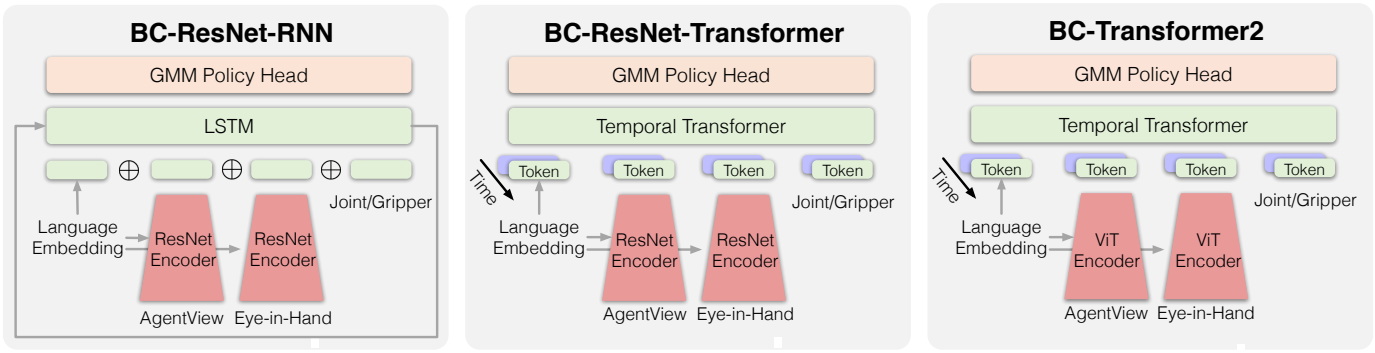


Fig. 3. We provide visualizations of the architectures for ResNet-RNN, ResNet-T, and ViT-T, respectively. It is worth noting that each model architecture incorporates language embedding in distinct ways.

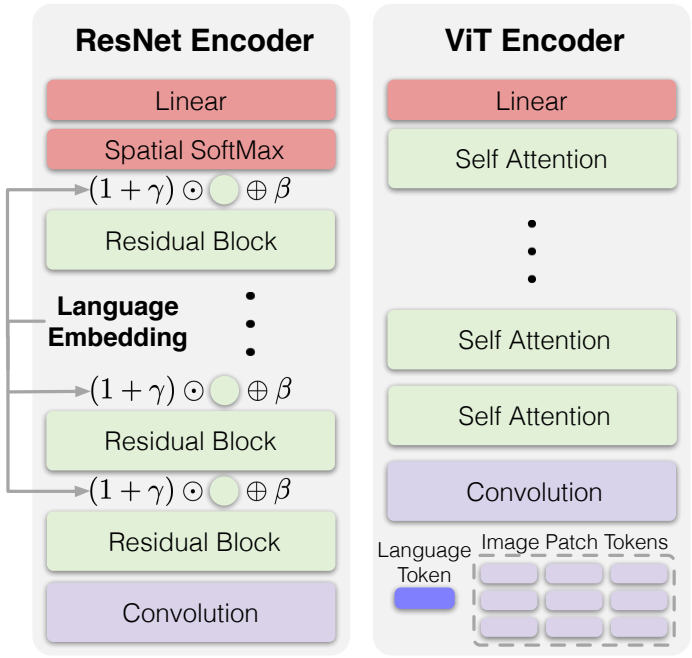


Fig. 4. The image encoders: ResNet-based encoder and the vision transformer-based encoder.

Variable	Value
resnet_image_embed_size	64
text_embed_size	32
rnn_hidden_size	1024
rnn_layer_num	2
rnn_dropout	0.0

TABLE II
HYPER PARAMETERS OF RESNET-RNN.

Variable	Value
extra_info_hidden_size	128
img_embed_size	64
transformer_num_layers	4
transformer_num_heads	6
transformer_head_output_size	64
transformer_mlp_hidden_size	256
transformer_dropout	0.1
transformer_max_seq_len	10

TABLE III
HYPER PARAMETERS OF RESNET-T.

Variable	Value
extra_info_hidden_size	128
img_embed_size	128
spatial_transformer_num_layers	7
spatial_transformer_num_heads	8
spatial_transformer_head_output_size	120
spatial_transformer_mlp_hidden_size	256
spatial_transformer_dropout	0.1
spatial_down_sample_embed_size	64
temporal_transformer_input_size	null
temporal_transformer_num_layers	4
temporal_transformer_num_heads	6
temporal_transformer_head_output_size	64
temporal_transformer_mlp_hidden_size	256
temporal_transformer_dropout	0.1
temporal_transformer_max_seq_len	10

TABLE IV
HYPER PARAMETERS OF VIT-T.

B. Lifelong Learning Algorithms

Lifelong learning (LL) is a field of study that aims to understand how an agent can continually acquire and retain knowledge over an infinite sequence of tasks without catastrophically forgetting previous knowledge. Recent literature proposes three main approaches to address the problem of catastrophic forgetting in deep learning.

The dynamic architecture approach gradually expands the learning model to incorporate new knowledge [14], [41]–[44]. Regularization-based methods, on the other hand, regularize the learner to a previous checkpoint when it learns a new task [7], [45], [46]. Rehearsal methods save exemplar data from prior tasks and replay them with new data to consolidate the agent's memory [13], [47]–[49]. For a comprehensive review of LL methods, we refer readers to surveys [50], [51].

The following paragraphs provide details on the three lifelong learning algorithms that we have implemented.

a) ER*:* Experience Replay (ER) [13] is a **rehearsal-based** approach that maintains a memory buffer of samples from previous tasks and leverages it to learn new tasks. After the completion of policy learning for a task, ER stores a portion of the data into a storage memory. When training a new task, ER samples data from the memory and combines it with the training data from the current task so that the training data approximately represents the empirical distribution of all-task data. In our implementation, we use a replay buffer to store a portion of the training data (up to 1000 trajectories) after training each task. For every training iteration during the training of a new task, we uniformly sample a fixed number of replay data from the memory (32 trajectories) along with each batch of training data from the new task.

b) EWC*:* Elastic Weight Consolidation(EWC) [7] is a **regularization-based** approach that add a regularization term that constraints neural network update to the original single-task learning objective. Specifically, EWC uses the Fisher information matrix that quantify the importance of every neural netwrk parameter. The loss function for task k is:

$$\mathcal{L}_k^{EWC}(\theta) = \mathcal{L}_K^{BC}(\theta) + \sum_i \frac{\lambda}{2} F_i \left(\theta_i - \theta^*_{k-1,i}\right)^2,$$

where λ is a penalty hyperparameter, and the coefficient F_i is the diagonal of the Fisher information matrix: $F_k = \mathbb{E}_{s\sim\mathcal{D}_k}\mathbb{E}_{a\sim p_\theta(\cdot|s)}\left(\nabla_{\theta_k}\log p_{\theta_k}(a|s)\right)^2$. In this work, we use the online update version of EWC that updates the Fisher information matrix using exponential moving average along the lifelong learning process, and use the empirical estimation of above Fisher information matrix to stabilize the estimation. Formally, the actually used estimation of Fisher Information Matrix is $\tilde{F}_k = \gamma F_{k-1} + (1-\gamma)F_k$, where $F_k = \mathbb{E}_{(s,a)\sim\mathcal{D}_k}\left(\nabla_{\theta_k}\log p_{\theta_k}(a|s)\right)^2$ and k is the task number. We set $\gamma = 0.9$ and $\lambda = 5\cdot 10^4$.

c) PACKNET*:* PACKNET [14] is a **dynamic architecture-based** approach that aims to prevent changes to parameters that are important for previous tasks in lifelong learning. To achieve this, PACKNET iteratively trains, prunes, fine-tunes, and freezes parts of the network. The method theoretically completely avoids catastrophic forgetting, but for each new task, the number of available parameters shrinks. The pruning process in PACKNET involves two stages. First, the network is trained, and at the end of the training, a fixed proportion of the most important parameters (25% in our implementation) are chosen, and the rest are pruned. Second, the selected part of the network is fine-tuned and then frozen. In our implementation, we follow the original paper [14] and do not train all biases and normalization layers. We perform the same number of fine-tuning epochs as for training (50 epochs in our implementation). Note that all evaluation metrics are calculated *before* the fine-tuning stage.

APPENDIX C
LIBERO TASK SUITE DESIGNS

A. Task Suites

We visualize all the tasks from the four task suites in Figure 5- 8. Figure 5 visualizes the initial states since the task goals are always the same. All the figures visualize the goal states of tasks except for Figure 5, which visualizes the initial states since the task goals are always the same.

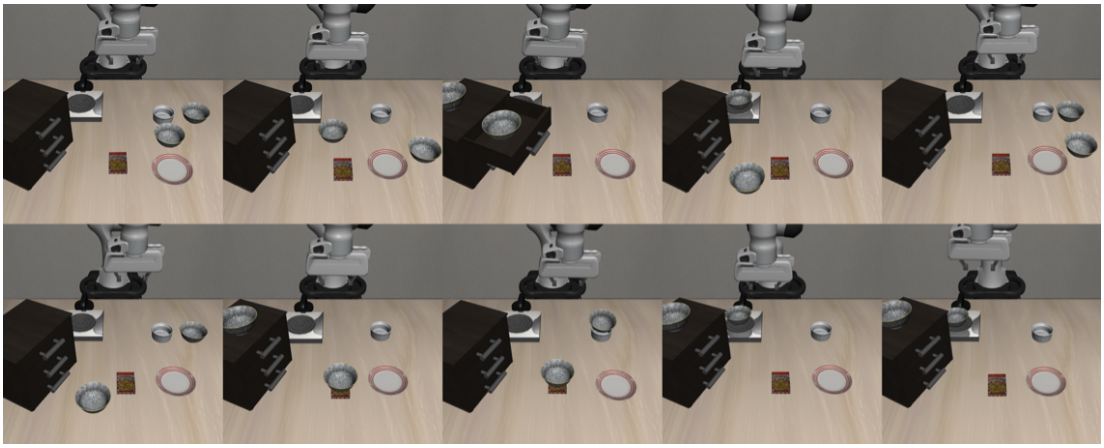


Fig. 5. LIBERO-SPATIAL

Fig. 6. LIBERO-OBJECT

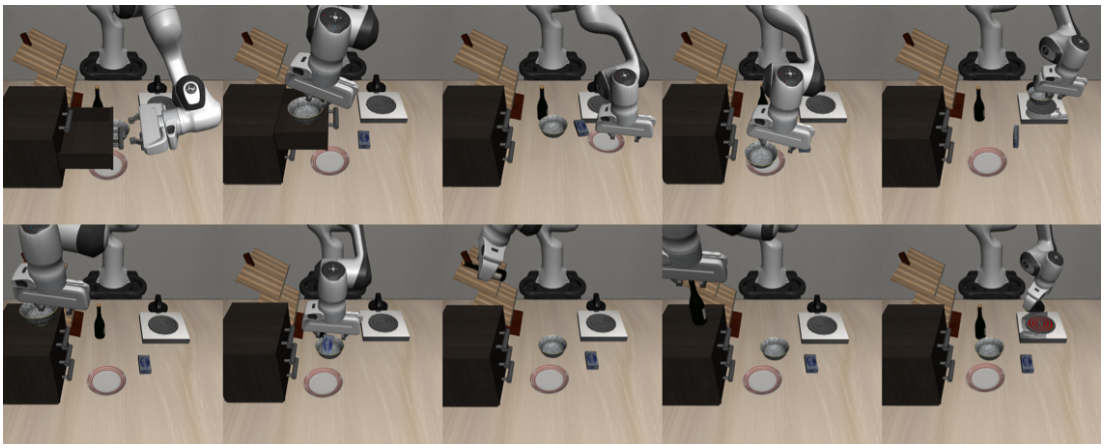


Fig. 7. LIBERO-GOAL

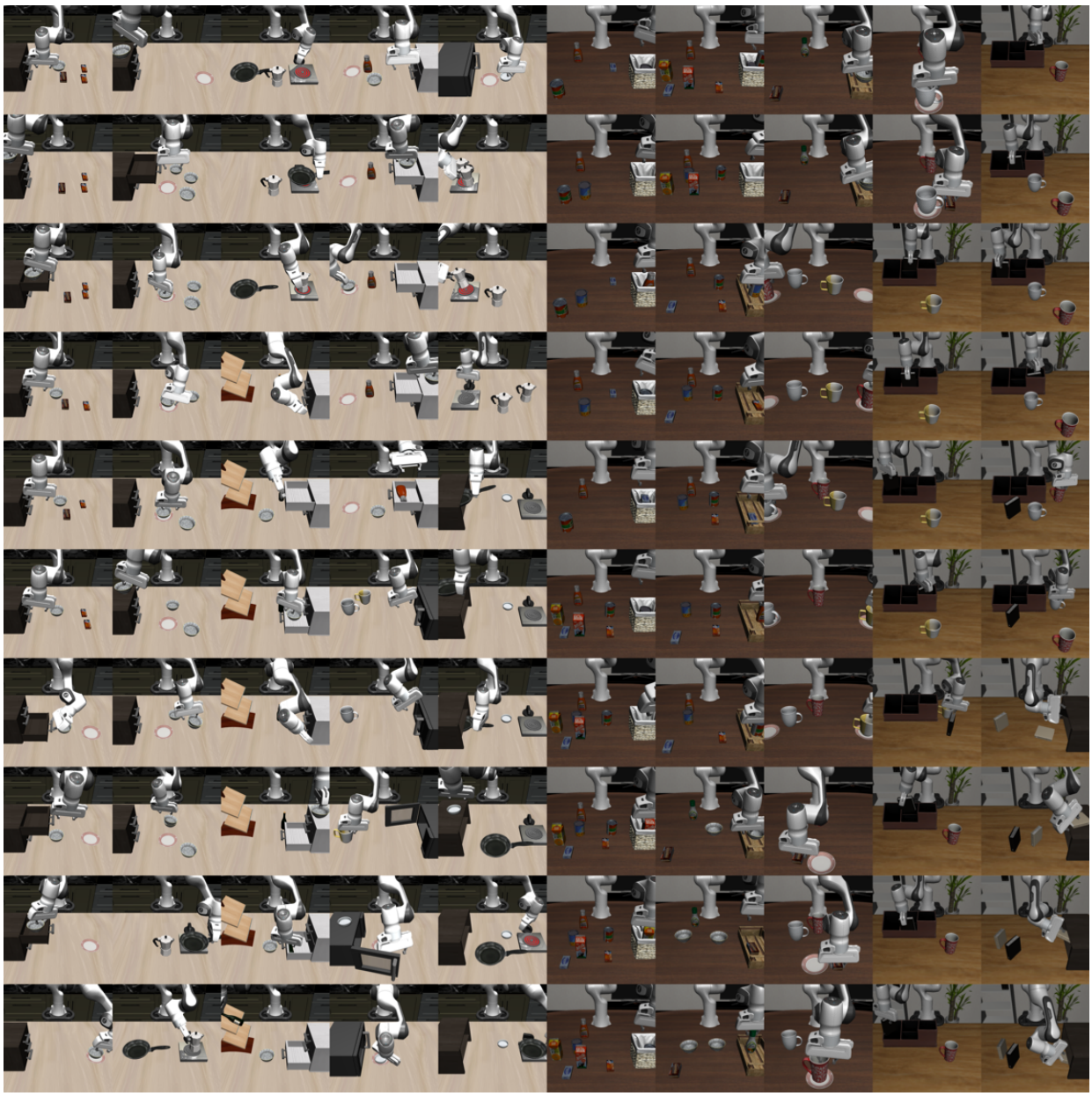


Fig. 8. LIBERO-100

B. PDDL-based Scene Description File

Here we visualize the whole content of an example scene description file based on PDDL. This file corresponds to the task shown in Figure 2.

Example task: *Open the top drawer of the cabinet and put the bowl in it.*

```
(define (problem LIBERO_Kitchen_Tabletop_Manipulation)
  (:domain robosuite)
  (:language open the top drawer of the cabinet and put the bowl in it)
    (:regions
      (wooden_cabinet_init_region
          (:target kitchen_table)
          (:ranges (
              (-0.01 -0.31 0.01 -0.29)
            )
          )
          (:yaw_rotation (
              (3.14159 3.14159)
            )
          )
      )
      (akita_black_bowl_init_region
          (:target kitchen_table)
          (:ranges (
              (-0.025 -0.025 0.025 0.025)
            )
          )
          (:yaw_rotation (
              (0.0 0.0)
            )
          )
      )
      (plate_init_region
          (:target kitchen_table)
          (:ranges (
              (-0.025 0.225 0.025 0.275)
            )
          )
          (:yaw_rotation (
              (0.0 0.0)
            )
          )
      )
      (top_side
          (:target wooden_cabinet_1)
      )
      (top_region
          (:target wooden_cabinet_1)
      )
      (middle_region
          (:target wooden_cabinet_1)
      )
      (bottom_region
          (:target wooden_cabinet_1)
      )
    )
```

```
  (:fixtures
    kitchen_table - kitchen_table
    wooden_cabinet_1 - wooden_cabinet
  )

  (:objects
    akita_black_bowl_1 - akita_black_bowl
    plate_1 - plate
  )

  (:obj_of_interest
    wooden_cabinet_1
    akita_black_bowl_1
  )

  (:init
    (On akita_black_bowl_1 kitchen_table_akita_black_bowl_init_region)
    (On plate_1 kitchen_table_plate_init_region)
    (On wooden_cabinet_1 kitchen_table_wooden_cabinet_init_region)
  )

  (:goal
    (And (Open wooden_cabinet_1_top_region)
         (In akita_black_bowl_1 wooden_cabinet_1_top_region)
    )
  )

)
```

APPENDIX D
EXPERIMENT

Experiments are conducted as an initial study for the five research topics mentioned in Section II. Specifically, we focus on addressing the following research questions:

- **Q1**: How do different neural policies/lifelong learning algorithms perform under specific distribution shifts?
- **Q2**: To what extent does neural architecture impact knowledge transfer in LLDM, and are there any discernible patterns in the specialized capabilities of each architecture?
- **Q3**: How do existing algorithms from the literature of lifelong supervised learning perform on LLDM tasks?
- **Q4**: To what extent does language embedding affect knowledge transfer in LLDM?
- **Q5**: How do different lifelong learning algorithms compare in terms of robustness to task ordering in LLDM?
- **Q6**: Can supervised pretraining improve downstream lifelong learning performance in LLDM?

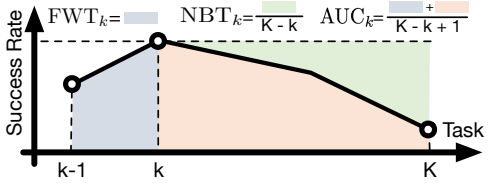
In the following, we first overview the experimental setup and introduce the evaluation metrics. Then we provide the empirical evaluations and findings for each of the research questions.

A. Experimental Setup

We consider five lifelong learning algorithms: SEQL the sequential learning baseline where the agent learns each task in the sequence directly without any further consideration, MTL the multitask learning baseline where the agent learns all tasks in the sequence simultaneously, the regularization-based method EWC [7], the replay-based method ER [13], and the dynamic architecture-based method PACKNET [14]. SEQL and MTL can be seen as approximations of the lower and upper bounds respectively for any lifelong learning algorithm. The other three methods represent the three primary categories of lifelong learning algorithms. For the neural architectures, we consider three vision-language policy architectures: RESNET-RNN, RESNET-T, VIT-T, which differ in how spatial or temporal information is aggregated (See Appendix B-A for more details). For each task, the agent is trained over 50 epochs on the 50 demonstration trajectories. We evaluate the agent's average success rate over 20 test rollout trajectories of a maximum length of 600 every 5 epochs. We use Adam optimizer [52] with a batch size of 32, and a cosine scheduled learning rate from 0.0001 to 0.00001 for each task. Following the convention of `Robomimic` [5], we pick the model checkpoint that achieves the best success rate as the final policy for a given task. After 50 epochs of training, the agent with the best checkpoint is then evaluated on all previously learned tasks, with 20 test rollout trajectories for each task. All policy networks are matched in Floating Point Operations Per Second (FLOPS): all policy architectures have ∼13.5G FLOPS. For each combination of algorithm, policy architecture, and task suite, we run the lifelong learning method 3 times with random seeds $\{100, 200, 300\}$ (180 experiments in total). See Table I for the implemented algorithms and architectures.

B. Evaluation Metrics

We report three metrics: FWT (forward transfer) [53], NBT (negative backward transfer), and AUC (area under the success rate curve). All metrics are computed in terms of success rate, as previous literature has shown that the success rate is a more reliable metric than training loss for manipulation policies [5] (Detailed explanation in Appendix D-F). Lower NBT means a policy has better performance in the previously seen tasks, higher FWT means a policy learns faster on a new task, and higher AUC means an overall better performance considering both NBT and FWT. Specifically, denote $c_{i,j,e}$ as the agent's success rate on task j when it learned over $i-1$ previous tasks and has just learned e epochs ($e \in \{0, 5, \ldots, 50\}$) on task i. Let $c_{i,i}$ be the best success rate over all evaluated epochs e for the current task i (i.e., $c_{i,i} = \max_e c_{i,i,e}$). Then, we find the earliest epoch e_i^* in which the agent achieves the best performance on task i (i.e., $e_i^* =_e c_{i,i,e_i} = c_{i,i}$), and assume for all $e \geq e_i^*$, $c_{i,i,e} = c_{i,i}$.[3] Given a different task $j \neq i$, we define $c_{i,j} = c_{i,j,e_i^*}$. Then the three metrics are defined as:



figureFWT$_k$ measures how fast the agent learns on task k. NBT$_k$ measures how much knowledge about task k the agent retains. AUC$_k$ summarizes both metrics for task k.

$$
\begin{aligned}
\text{FWT} &= \sum_{k\in[K]} \frac{\text{FWT}_k}{K}, \quad \text{FWT}_k = \frac{1}{11} \sum_{e\in\{0\ldots50\}} c_{k,k,e} \\
\text{NBT} &= \sum_{k\in[K]} \frac{\text{NBT}_k}{K}, \quad \text{NBT}_k = \frac{1}{K-k} \sum_{\tau=k+1}^{K} \big(c_{k,k} - c_{\tau,k}\big) \\
\text{AUC} &= \sum_{k\in[K]} \frac{\text{AUC}_k}{K}, \quad \text{AUC}_k = \frac{1}{K-k+1} \big(\text{FWT}_k + \sum_{\tau=k+1}^{K} c_{\tau,k}\big)
\end{aligned}
\tag{1}
$$

[3]In practice, it's possible that the agent's performance on task i is not monotonically increasing due to the variance of learning. But we keep the best checkpoint among those saved at epochs $\{e\}$ as if the agent stops learning after e_i^*.

C. Experimental Results

We present empirical results to address the research questions. Please refer to Appendix D-D for the full results across all algorithms, policy architectures, and task suites.

*a) Study on the Policy's Neural Architectures (**Q1**, **Q2**) :* Table V reports the agent's lifelong learning performance using the three different neural architectures on the four task suites. Results are reported when ER and PackNet are used as they demonstrate the best lifelong learning performance across all task suites.

Policy Arch.	ER			PackNet		
	FWT(↑)	NBT(↓)	AUC(↑)	FWT(↑)	NBT(↓)	AUC(↑)
	LIBERO-Long					
ResNet-RNN	0.16 ± 0.02	**0.16** ± 0.02	0.08 ± 0.01	0.13 ± 0.00	0.21 ± 0.01	0.03 ± 0.00
ResNet-T	**0.48** ± 0.02	0.32 ± 0.04	**0.32** ± 0.01	0.22 ± 0.01	**0.08** ± 0.01	0.25 ± 0.00
ViT-T	0.38 ± 0.05	0.29 ± 0.06	0.25 ± 0.02	**0.36** ± 0.01	0.14 ± 0.01	**0.34** ± 0.01
	LIBERO-Spatial					
ResNet-RNN	0.40 ± 0.02	0.29 ± 0.02	0.29 ± 0.01	0.27 ± 0.03	0.38 ± 0.03	0.06 ± 0.01
ResNet-T	**0.65** ± 0.03	**0.27** ± 0.03	**0.56** ± 0.01	0.55 ± 0.01	**0.07** ± 0.02	**0.63** ± 0.00
ViT-T	0.63 ± 0.01	0.29 ± 0.02	0.50 ± 0.02	**0.57** ± 0.04	0.15 ± 0.00	0.59 ± 0.03
	LIBERO-Object					
ResNet-RNN	0.30 ± 0.01	**0.27** ± 0.05	0.17 ± 0.05	0.29 ± 0.02	0.35 ± 0.02	0.13 ± 0.01
ResNet-T	0.67 ± 0.07	0.43 ± 0.04	0.44 ± 0.06	**0.60** ± 0.07	**0.17** ± 0.05	**0.60** ± 0.05
ViT-T	**0.70** ± 0.02	0.28 ± 0.01	**0.57** ± 0.01	0.58 ± 0.03	0.18 ± 0.02	0.56 ± 0.04
	LIBERO-Goal					
ResNet-RNN	0.41 ± 0.00	0.35 ± 0.01	0.26 ± 0.01	0.32 ± 0.03	0.37 ± 0.04	0.11 ± 0.01
ResNet-T	**0.64** ± 0.01	**0.34** ± 0.02	**0.49** ± 0.02	0.63 ± 0.02	**0.06** ± 0.01	0.75 ± 0.01
ViT-T	0.57 ± 0.00	0.40 ± 0.02	0.38 ± 0.01	**0.69** ± 0.02	0.08 ± 0.01	**0.76** ± 0.02

TABLE V

PERFORMANCE OF THE THREE NEURAL ARCHITECTURES USING ER AND PACKNET ON THE FOUR TASK SUITES. RESULTS ARE AVERAGED OVER THREE SEEDS AND WE REPORT THE MEAN AND STANDARD ERROR. THE BEST PERFORMANCE IS **BOLDED**, AND COLORED IN **PURPLE** IF THE IMPROVEMENT IS STATISTICALLY SIGNIFICANT OVER OTHER NEURAL ARCHITECTURES, WHEN A TWO-TAILED, STUDENT'S T-TEST UNDER EQUAL SAMPLE SIZES AND UNEQUAL VARIANCE IS APPLIED WITH A p-VALUE OF 0.05.

Findings: First, we observe that ResNet-T and ViT-T work much better than ResNet-RNN on average, indicating that using a transformer on the "temporal" level could be a better option than using an RNN model. Second, the performance difference among different architectures depends on the underlying lifelong learning algorithm. If PackNet (a dynamic architecture approach) is used, we observe no significant performance difference between ResNet-T and ViT-T except on the LIBERO-Long task suite where ViT-T performs much better than ResNet-T. In contrast, if ER is used, we observe that ResNet-T performs better than ViT-T on all task suites except LIBERO-Object. This potentially indicates that the ViT architecture is better at processing visual information with more object varieties than the ResNet architecture when the network capacity is sufficiently large (See the MTL results in Table VIII on LIBERO-Object as the supporting evidence). The above findings shed light on how one can improve architecture design for better processing of spatial and temporal information in LLDM.

Study on Lifelong Learning Algorithms (**Q1**, **Q3**) Table VI reports the lifelong learning performance of the three lifelong learning algorithms, together with the SeqL and MTL baselines. All experiments use the same ResNet-T architecture as it performs the best across all policy architectures.

Findings: We observed a series of interesting findings that could potentially benefit future research on algorithm design for LLDM: **1)** SeqL shows the best FWT over all task suites. This is surprising since it indicates all lifelong learning algorithms we consider actually hurt forward transfer; **2)** PackNet outperforms other lifelong learning algorithms on LIBERO-X but is outperformed by ER significantly on LIBERO-Long, mainly because of low forward transfer. This confirms that the dynamic architecture approach is good at preventing forgetting. But since PackNet splits the network into different sub-networks, the essential capacity of the network for learning any individual task is smaller. Therefore, we conjecture that PackNet is not rich enough to learn on LIBERO-Long; **3)** EWC works worse than SeqL, showing that the regularization on the loss term can actually impede the agent's performance on LLDM problems (See Appendix D-F); and **4)** ER, the rehearsal method, is robust across all task suites.

Study on Language Embeddings as the Task Identifier (**Q4**) To investigate to what extent language embedding play a role in LLDM, we compare the performance of the same lifelong learner using four different pretrained language embeddings. Namely, we choose BERT [15], CLIP [54], GPT-2 [55] and the Task-ID embedding. Task-ID embeddings are produced by feeding a string such as "Task 5" into a pretrained BERT model.

Lifelong Algo.	FWT(↑)	NBT(↓)	AUC(↑)	FWT(↑)	NBT(↓)	AUC(↑)
	LIBERO-LONG			LIBERO-SPATIAL		
SEQL	**0.54** ± 0.01	0.63 ± 0.01	0.15 ± 0.00	**0.72** ± 0.01	0.81 ± 0.01	0.20 ± 0.01
ER	0.48 ± 0.02	0.32 ± 0.04	**0.32** ± 0.01	0.65 ± 0.03	0.27 ± 0.03	0.56 ± 0.01
EWC	0.13 ± 0.02	0.22 ± 0.03	0.02 ± 0.00	0.23 ± 0.01	0.33 ± 0.01	0.06 ± 0.01
PACKNET	0.22 ± 0.01	**0.08** ± 0.01	0.25 ± 0.00	0.55 ± 0.01	**0.07** ± 0.02	**0.63** ± 0.00
MTL			0.48 ± 0.01			0.83 ± 0.00
	LIBERO-OBJECT			LIBERO-GOAL		
SEQL	**0.78** ± 0.04	0.76 ± 0.04	0.26 ± 0.02	**0.77** ± 0.01	0.82 ± 0.01	0.22 ± 0.00
ER	0.67 ± 0.07	0.43 ± 0.04	0.44 ± 0.06	0.64 ± 0.01	0.34 ± 0.02	0.49 ± 0.02
EWC	0.56 ± 0.03	0.69 ± 0.02	0.16 ± 0.02	0.32 ± 0.02	0.48 ± 0.03	0.06 ± 0.00
PACKNET	0.60 ± 0.07	**0.17** ± 0.05	**0.60** ± 0.05	0.63 ± 0.02	**0.06** ± 0.01	**0.75** ± 0.01
MTL			0.54 ± 0.02			0.80 ± 0.01

TABLE VI

PERFORMANCE OF THREE LIFELONG ALGORITHMS AND THE SEQL AND MTL BASELINES ON THE FOUR TASK SUITES, WHERE THE POLICY IS FIXED TO BE RESNET-T. RESULTS ARE AVERAGED OVER THREE SEEDS AND WE REPORT THE MEAN AND STANDARD ERROR. THE BEST PERFORMANCE IS **BOLDED**, AND COLORED IN **PURPLE** IF THE IMPROVEMENT IS STATISTICALLY SIGNIFICANT OVER OTHER ALGORITHMS, WHEN A TWO-TAILED, STUDENT'S T-TEST UNDER EQUAL SAMPLE SIZES AND UNEQUAL VARIANCE IS APPLIED WITH A p-VALUE OF 0.05.

Embedding Type	Dimension	FWT(↑)	NBT(↓)	AUC(↑)
BERT	768	0.48 ± 0.02	**0.32** ± 0.04	0.32 ± 0.01
CLIP	512	**0.52** ± 0.00	0.34 ± 0.01	**0.35** ± 0.01
GPT-2	768	0.46 ± 0.01	0.34 ± 0.02	0.30 ± 0.01
Task-ID	768	0.50 ± 0.01	0.37 ± 0.01	0.33 ± 0.01

TABLE VII

PERFORMANCE OF A LIFELONG LEARNER USING FOUR DIFFERENT LANGUAGE EMBEDDINGS ON LIBERO-LONG, WHERE WE FIX THE POLICY ARCHITECTURE TO RESNET-T AND THE LIFELONG LEARNING ALGORITHM TO ER. THE TASK-ID EMBEDDINGS ARE RETRIEVED BY FEEDING "TASK + ID" INTO A PRETRAINED BERT MODEL. RESULTS ARE AVERAGED OVER THREE SEEDS AND WE REPORT THE MEAN AND STANDARD ERROR. THE BEST PERFORMANCE IS **BOLDED**. NOTE THAT NO STATISTICALLY SIGNIFICANT DIFFERENCE IS OBSERVED AMONG THE DIFFERENT LANGUAGE EMBEDDINGS.

Findings: From Table VII, we observe that *no* statistically significant difference exists among the different language embeddings. More interestingly, we find that the Task-ID embedding is equally competitive against other embeddings that are supposed to possess richer semantic meaning about the task. Such findings suggest there exists much room for improving the performance by more explicitly leveraging the semantic information inside the task description. As there exists no statistically significant difference, we choose BERT embeddings as the default task embedding.

Study on Task Ordering (**Q5**) We apply ER and PACKNET to five different task orderings on LIBERO-LONG, to investigate how robust these methods are to the task ordering.

Findings: From Figure 16, we observe that indeed different task ordering could result in very different performances for the same algorithm. Specifically, such difference is statistically significant for PACKNET.

Study on How Pretraining Affects Downstream LLDM (**Q6**) Fig 9 reports the results on LIBERO-LONG of five combinations of algorithms and policy architectures, when the underlying model is pretrained on the 90 short-horizion tasks in LIBERO-100 or learned from scratch. For pretraining, we apply behavioral cloning on the 90 tasks using the three policy architectures for 50 epochs. We save a checkpoint every 5 epochs of training and then pick the checkpoint for each architecture that has the best performance as the pretrained model for downstream LLDM.

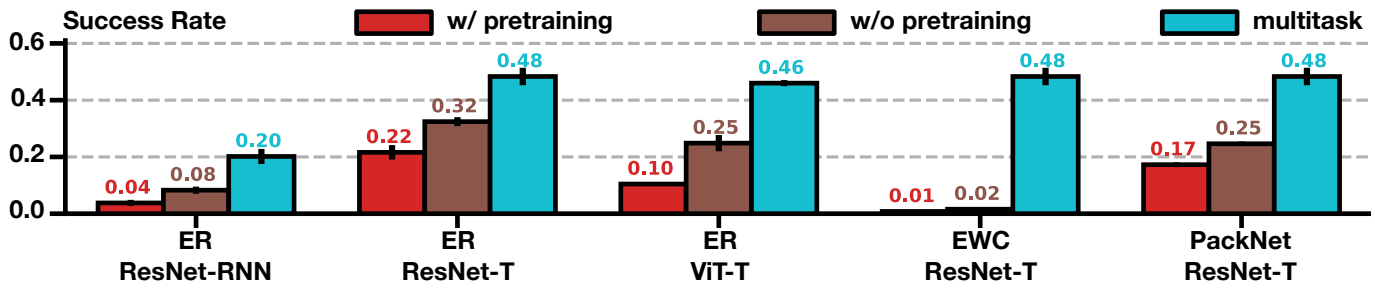


Fig. 9. Performance of different combinations of algorithms and architectures without pretraining, with pretraining for (T5). We also include the multi-task learning performance for each architecture for reference.

Findings: We observe that the basic supervised pretraining can *hurt* the model's downstream lifelong learning performance. This, together with the results seen in Table VI (e.g., naive sequential fine-tuning has better forward transfer than when lifelong learning algorithms are applied), indicates that better pretraining techniques are needed.

b) Attention Visualization:: To better understand what type of knowledge the agent forgets during the lifelong learning process, we visualize the agent's attention map on each observed image input. The visualized saliency maps and the discussion can be found in Appendix D-G.

D. Full Results

We provide the full results across three different lifelong learning algorithms (e.g., EWC, ER, PACKNET) and three different policy architectures (e.g., RESNET-RNN, RESNET-T, VIT-T) on the four task suites in Table VIII.

Algo.	Policy Arch.	FWT(↑)	NBT(↓)	AUC(↑)	FWT(↑)	NBT(↓)	AUC(↑)
		LIBERO-LONG			LIBERO-SPATIAL		
SEQL	RESNET-RNN	0.24 ± 0.02	0.28 ± 0.01	0.07 ± 0.01	0.50 ± 0.01	0.61 ± 0.01	0.14 ± 0.01
	RESNET-T	0.54 ± 0.01	0.63 ± 0.01	0.15 ± 0.00	0.72 ± 0.01	0.81 ± 0.01	0.20 ± 0.01
	VIT-T	0.44 ± 0.04	0.50 ± 0.05	0.13 ± 0.01	0.63 ± 0.02	0.76 ± 0.01	0.16 ± 0.01
ER	RESNET-RNN	0.16 ± 0.02	0.16 ± 0.02	0.08 ± 0.01	0.40 ± 0.02	0.29 ± 0.02	0.29 ± 0.01
	RESNET-T	0.48 ± 0.02	0.32 ± 0.04	0.32 ± 0.01	0.65 ± 0.03	0.27 ± 0.03	0.56 ± 0.01
	VIT-T	0.38 ± 0.05	0.29 ± 0.06	0.25 ± 0.02	0.63 ± 0.01	0.29 ± 0.02	0.50 ± 0.02
EWC	RESNET-RNN	0.02 ± 0.00	0.04 ± 0.01	0.00 ± 0.00	0.14 ± 0.02	0.23 ± 0.02	0.03 ± 0.00
	RESNET-T	0.13 ± 0.02	0.22 ± 0.03	0.02 ± 0.00	0.23 ± 0.01	0.33 ± 0.01	0.06 ± 0.01
	VIT-T	0.05 ± 0.02	0.09 ± 0.03	0.01 ± 0.00	0.32 ± 0.03	0.48 ± 0.03	0.06 ± 0.01
PACKNET	RESNET-RNN	0.13 ± 0.00	0.21 ± 0.01	0.03 ± 0.00	0.27 ± 0.03	0.38 ± 0.03	0.06 ± 0.01
	RESNET-T	0.22 ± 0.01	0.08 ± 0.01	0.25 ± 0.00	0.55 ± 0.01	0.07 ± 0.02	0.63 ± 0.00
	VIT-T	0.36 ± 0.01	0.14 ± 0.01	0.34 ± 0.01	0.57 ± 0.04	0.15 ± 0.00	0.59 ± 0.03
MTL	RESNET-RNN			0.20 ± 0.01			0.61 ± 0.00
	RESNET-T			0.48 ± 0.01			0.83 ± 0.00
	VIT-T			0.46 ± 0.00			0.79 ± 0.01
		LIBERO-OBJECT			LIBERO-GOAL		
SEQL	RESNET-RNN	0.48 ± 0.03	0.53 ± 0.04	0.15 ± 0.01	0.61 ± 0.01	0.73 ± 0.01	0.16 ± 0.00
	RESNET-T	0.78 ± 0.04	0.76 ± 0.04	0.26 ± 0.02	0.77 ± 0.01	0.82 ± 0.01	0.22 ± 0.00
	VIT-T	0.76 ± 0.03	0.73 ± 0.03	0.27 ± 0.02	0.75 ± 0.01	0.85 ± 0.01	0.20 ± 0.01
ER	RESNET-RNN	0.30 ± 0.01	0.27 ± 0.05	0.17 ± 0.05	0.41 ± 0.00	0.35 ± 0.01	0.26 ± 0.01
	RESNET-T	0.67 ± 0.07	0.43 ± 0.04	0.44 ± 0.06	0.64 ± 0.01	0.34 ± 0.02	0.49 ± 0.02
	VIT-T	0.70 ± 0.02	0.28 ± 0.01	0.57 ± 0.01	0.57 ± 0.00	0.40 ± 0.02	0.38 ± 0.01
EWC	RESNET-RNN	0.17 ± 0.04	0.23 ± 0.04	0.06 ± 0.01	0.16 ± 0.01	0.22 ± 0.01	0.06 ± 0.01
	RESNET-T	0.56 ± 0.03	0.69 ± 0.02	0.16 ± 0.02	0.32 ± 0.02	0.48 ± 0.03	0.06 ± 0.00
	VIT-T	0.57 ± 0.03	0.64 ± 0.03	0.23 ± 0.00	0.32 ± 0.04	0.45 ± 0.04	0.07 ± 0.01
PACKNET	RESNET-RNN	0.29 ± 0.02	0.35 ± 0.02	0.13 ± 0.01	0.32 ± 0.03	0.37 ± 0.04	0.11 ± 0.01
	RESNET-T	0.60 ± 0.07	0.17 ± 0.05	0.60 ± 0.05	0.63 ± 0.02	0.06 ± 0.01	0.75 ± 0.01
	VIT-T	0.58 ± 0.03	0.18 ± 0.02	0.56 ± 0.04	0.69 ± 0.02	0.08 ± 0.01	0.76 ± 0.02
MTL	RESNET-RNN			0.10 ± 0.03			0.59 ± 0.00
	RESNET-T			0.54 ± 0.02			0.80 ± 0.01
	VIT-T			0.78 ± 0.02			0.82 ± 0.01

TABLE VIII

WE PRESENT THE FULL RESULTS OF ALL NETWORKS AND ALGORITHMS ON ALL FOUR TASK SUITES. FOR EACH TASK SUITE, WE HIGHLIGHT THE TOP THREE AUC SCORES AMONG THE COMBINATIONS OF THE THREE LIFELONG LEARNING ALGORITHMS AND THE THREE NEURAL ARCHITECTURES. THE BEST THREE RESULTS ARE HIGHLIGHTED IN MAGENTA (THE BEST), LIGHT MAGENTA (THE SECOND BEST), AND SUPER LIGHT MAGENTA (THE THIRD BEST), RESPECTIVELY.

To better illustrate the performance of each lifelong learning agent throughout the learning process, we present plots that show how the agent's performance evolves over the stream of tasks. Firstly, we provide plots that compare the performance of the agent using different lifelong learning algorithms while fixing the policy architecture (refer to Figure 10,11, and 12).

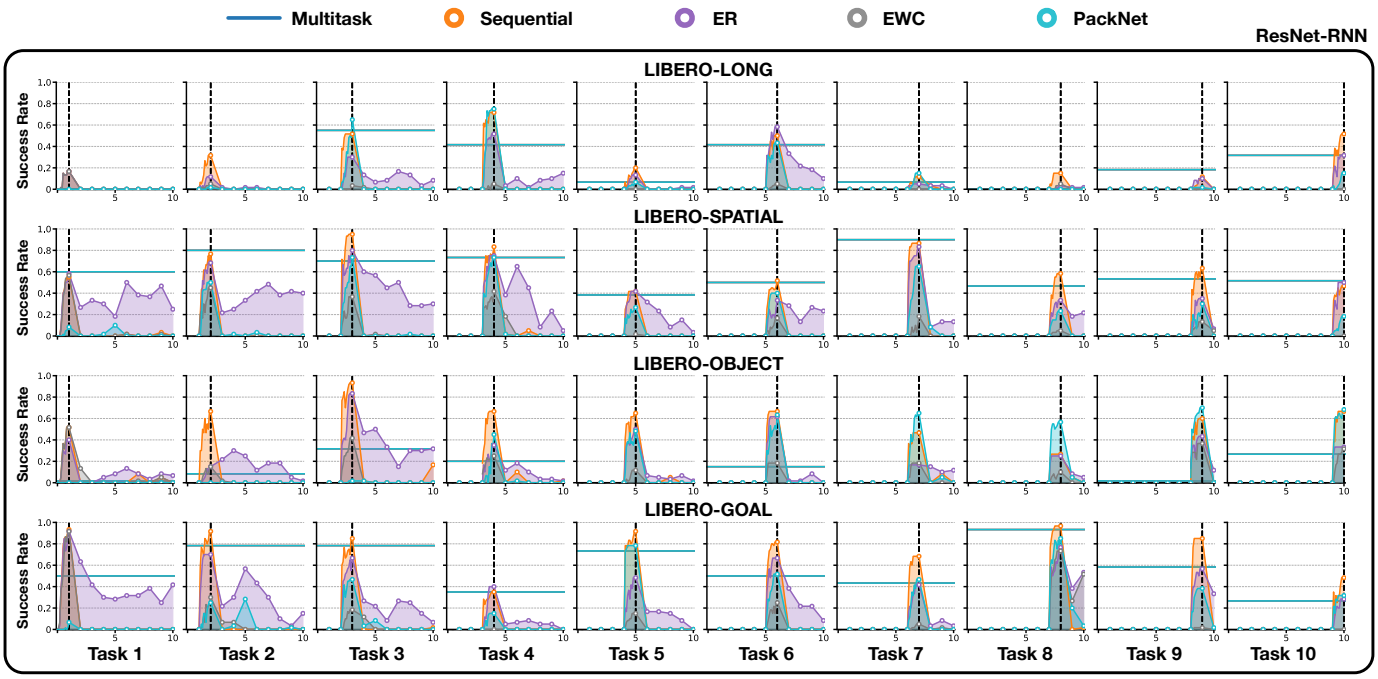


Fig. 10. We compare the performance of different algorithms using the ResNet-RNN policy architecture in Figure 10. The y-axis represents the success rate, and the x-axis shows the agent's performance on each of the 10 tasks in a specific task suite over the course of learning. For example, the upper-left plot in the figure displays the agent's performance on the first task as it learns the 10 tasks sequentially.

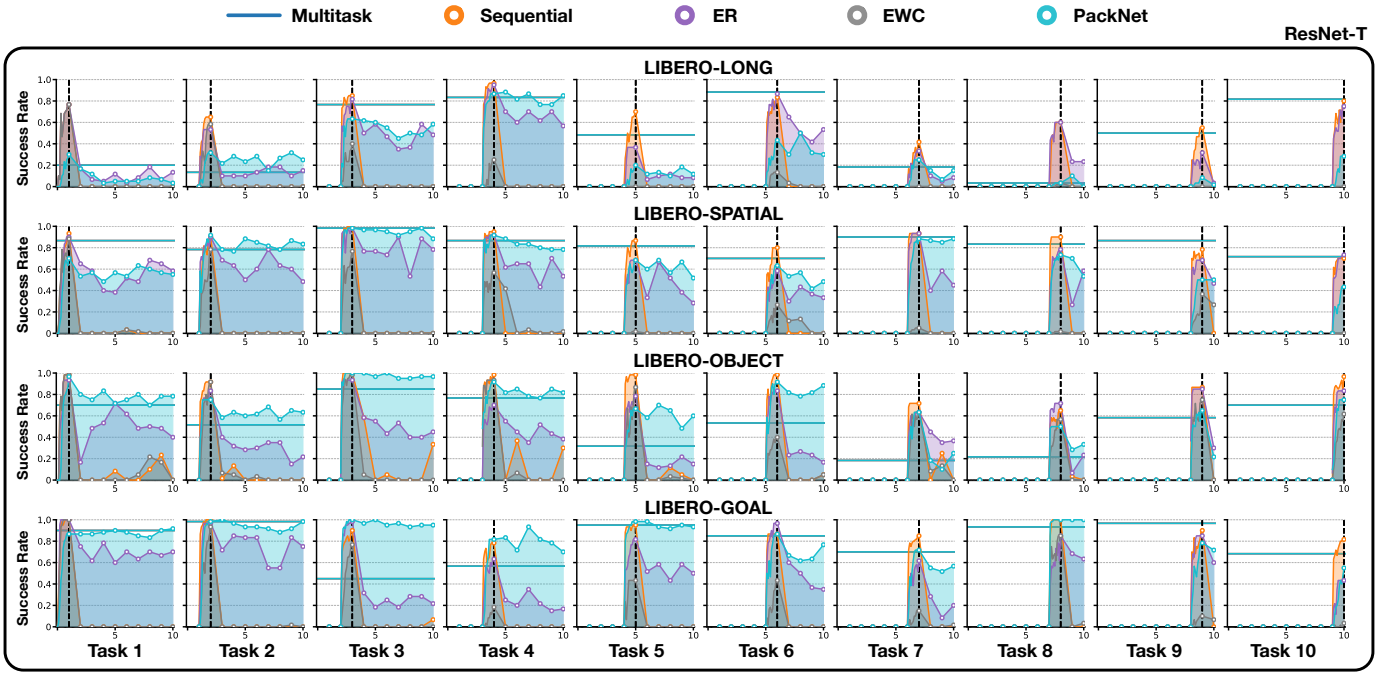


Fig. 11. Comparison of different algorithms using the ResNet-T policy architecture. The y-axis represents the success rate, while the x-axis shows the agent's performance on each of the 10 tasks in a given task suite during the course of learning. For example, the plot in the upper-left corner depicts the agent's performance on the first task as it learns the 10 tasks sequentially.

Next, we provide plots that compare the performance of the agent using different policy architectures while fixing the lifelong learning algorithm (refer to Figure13, 14, and 15)

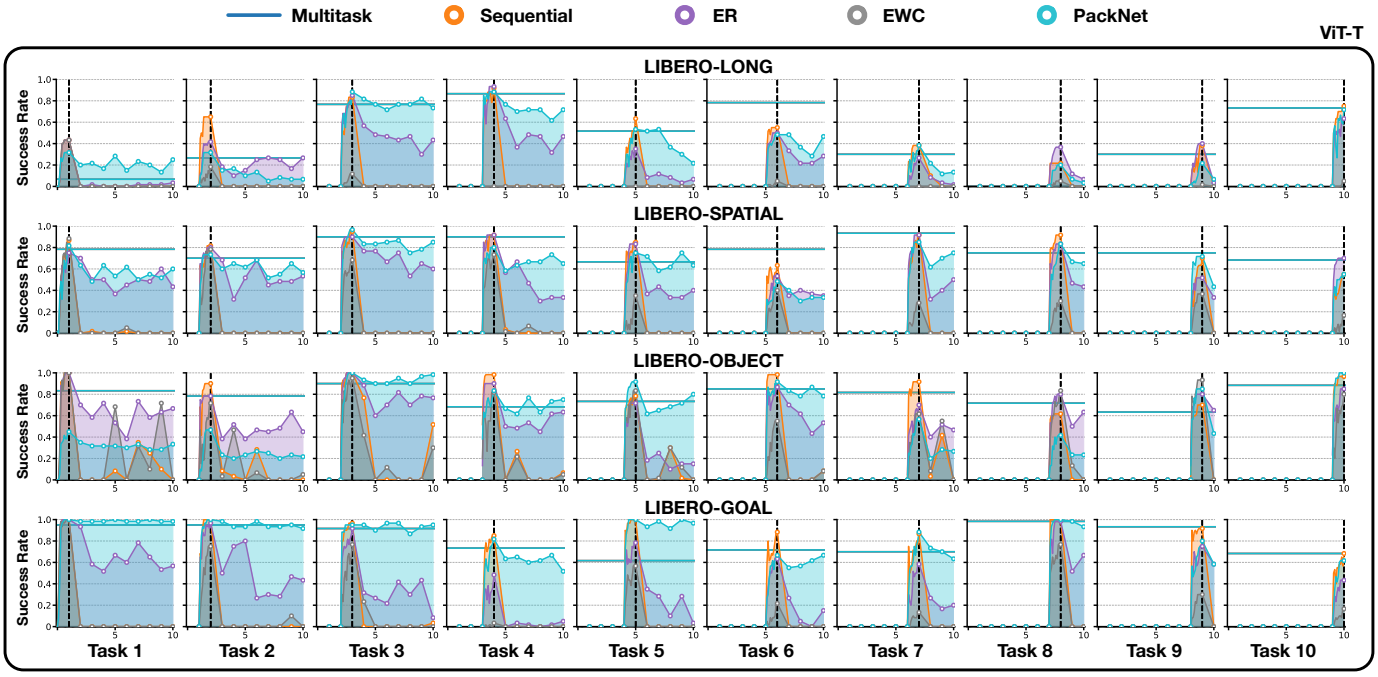


Fig. 12. Comparison of different algorithms using the ViT-T policy architecture. The success rate is represented on the y-axis, while the x-axis shows the agent's performance on the 10 tasks in a given task suite over the course of learning. For instance, the plot in the upper-left corner illustrates the agent's performance on the first task when learning the 10 tasks sequentially.

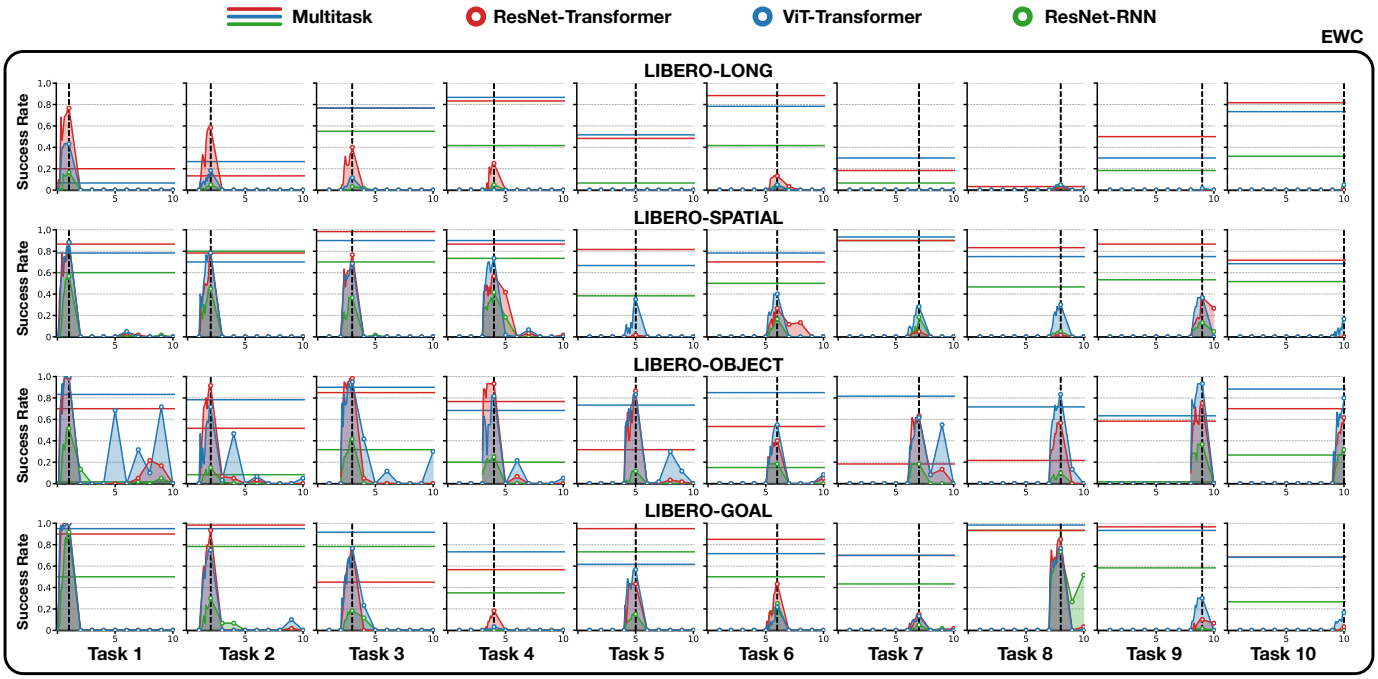


Fig. 13. Comparison of different architectures with the EWC algorithm. The y-axis is the success rate, while the x-axis shows the agent's performance on the 10 tasks in a given task suite over the course of learning. For instance, the upper-left plot shows the agent's performance on the first task when learning the 10 tasks sequentially.

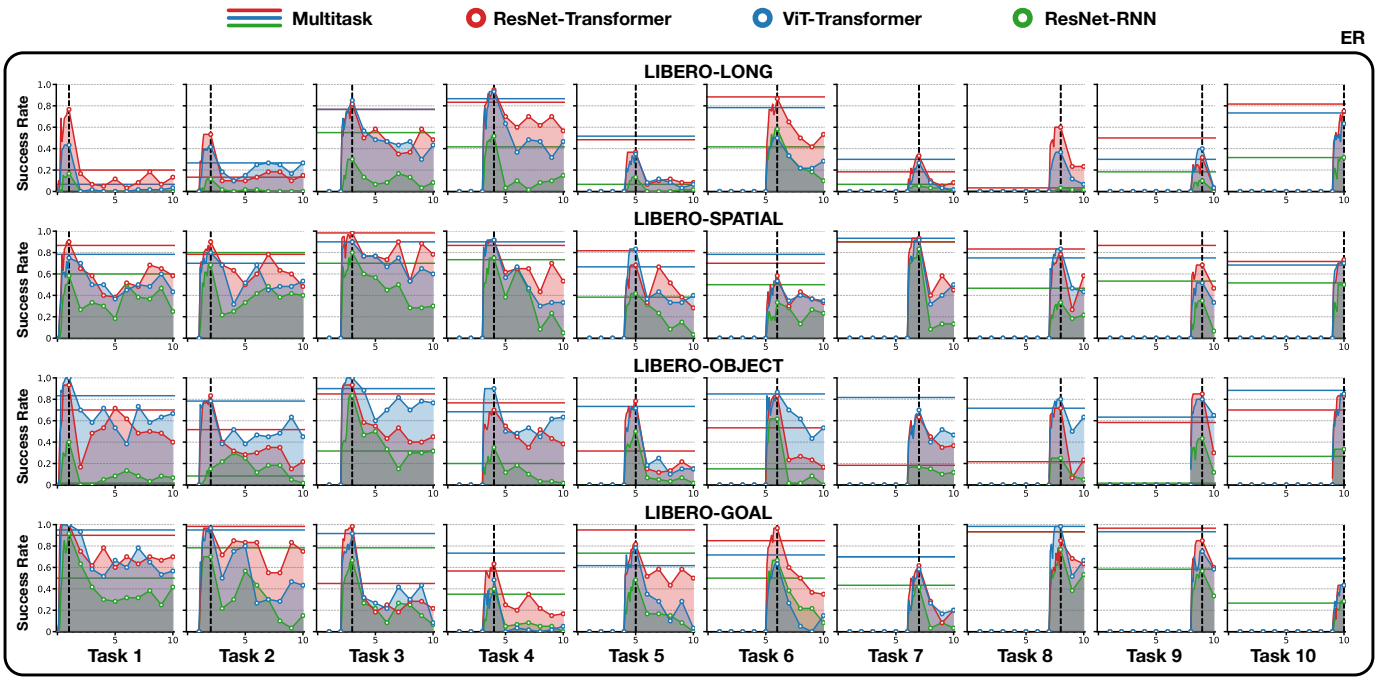


Fig. 14. Comparison of different architectures with the ER algorithm. The y-axis is the success rate, while the x-axis shows the agent's performance on the 10 tasks in a given task suite ver the course of learning. For instance, the upper-left plot shows the agent's performance on the first task when learning the 10 tasks sequentially.

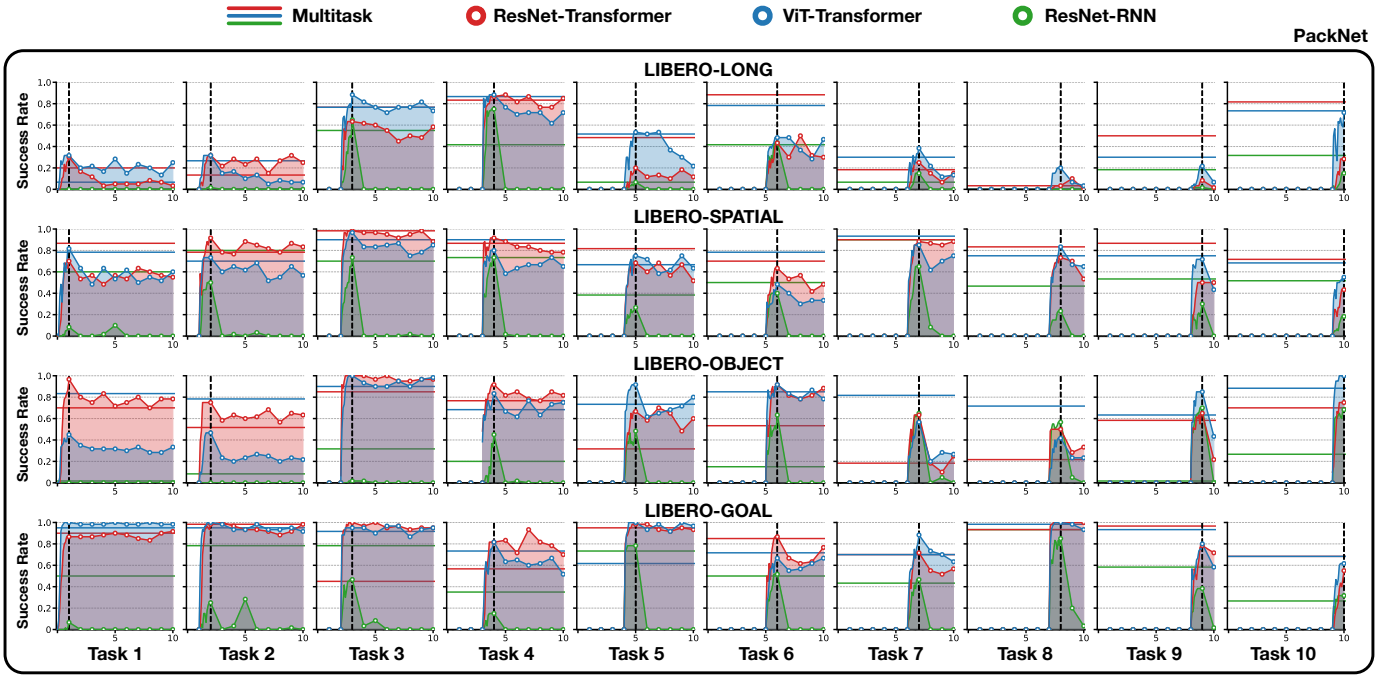


Fig. 15. Comparison of different architectures with the PackNet algorithm. The y-axis is the success rate, while the x-axis shows the agent's performance on the 10 tasks in a given task suite over the course of learning. For instance, the upper-left plot shows the agent's performance on the first task when learning the 10 tasks sequentially.

E. Study on task ordering (**Q4**)

Figure 16 shows the result of the study on **Q4**. For all experiments in this study, we used RESNET-T as the neural architecture and evaluated both ER and PACKNET. As the figure illustrates, the performance of both algorithms varies across different task orderings. This finding highlights an important direction for future research: developing algorithms or architectures that are robust to varying task orderings.

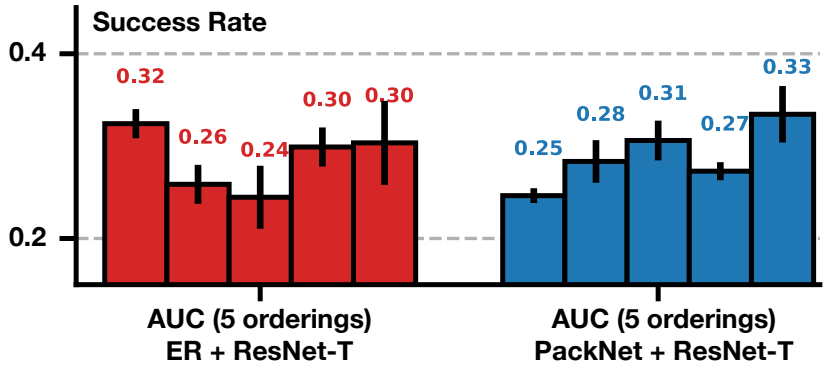


Fig. 16. Performance of ER and PACKNET using RESNET-T on five different task orderings. An error bar shows the performance standard deviation for a fixed ordering.

Findings: From Figure 16, we observe that indeed different task ordering could result in very different performances for the same algorithm. Specifically, such difference is statistically significant for PACKNET.

F. Loss v.s. Success Rates

We demonstrate that behavioral cloning loss can be a misleading indicator of task success rate in this section. In supervised learning tasks like image classifications, lower loss often indicates better prediction accuracy. However, this is not, in general, true for decision-making tasks. This is because errors can compound until failures during executing a robot [56]. Figure 17, 11 and 12 plots the training loss and success rates of three lifelong learning methods (ER, EWC, and PACKNET) for comparison. We evaluate the three algorithms on four task suites using three different neural architectures.

Findings: We observe that though sometimes EWC has the **lowest** loss, it did not achieve good success rate. ER, on the other hand, can have the highest loss but perform better than EWC. In conclusion, success rates, instead of behavioral cloning loss, should be the right metric to evaluate whether a model checkpoint is good or not.

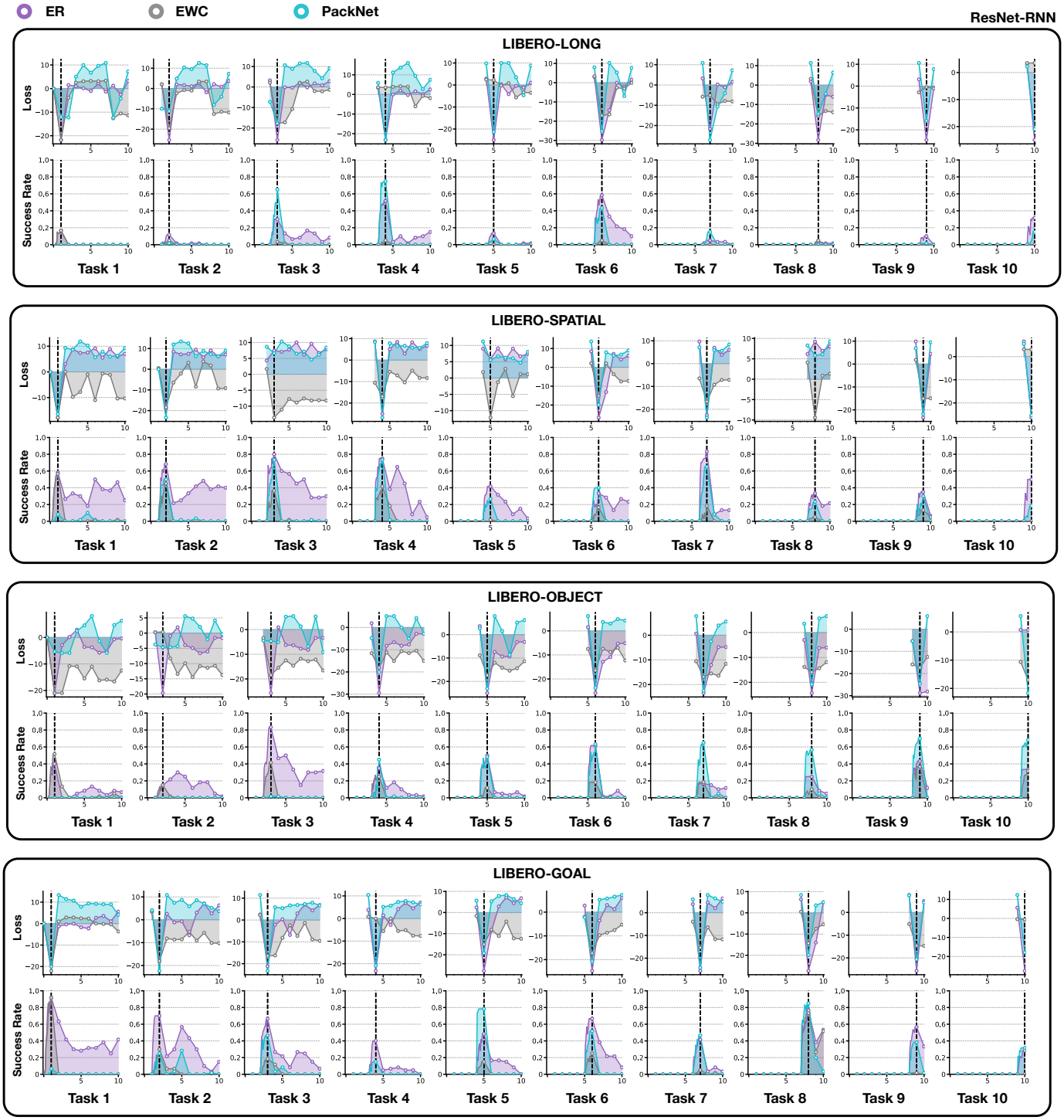


Fig. 17. Losses and success rates of ER (violet), EWC (grey), and PackNet (blue) on four task suites with ResNet-RNN policy. The first (second) row shows the loss (success rate) of the agent on task i throughout the LLDM procedure.

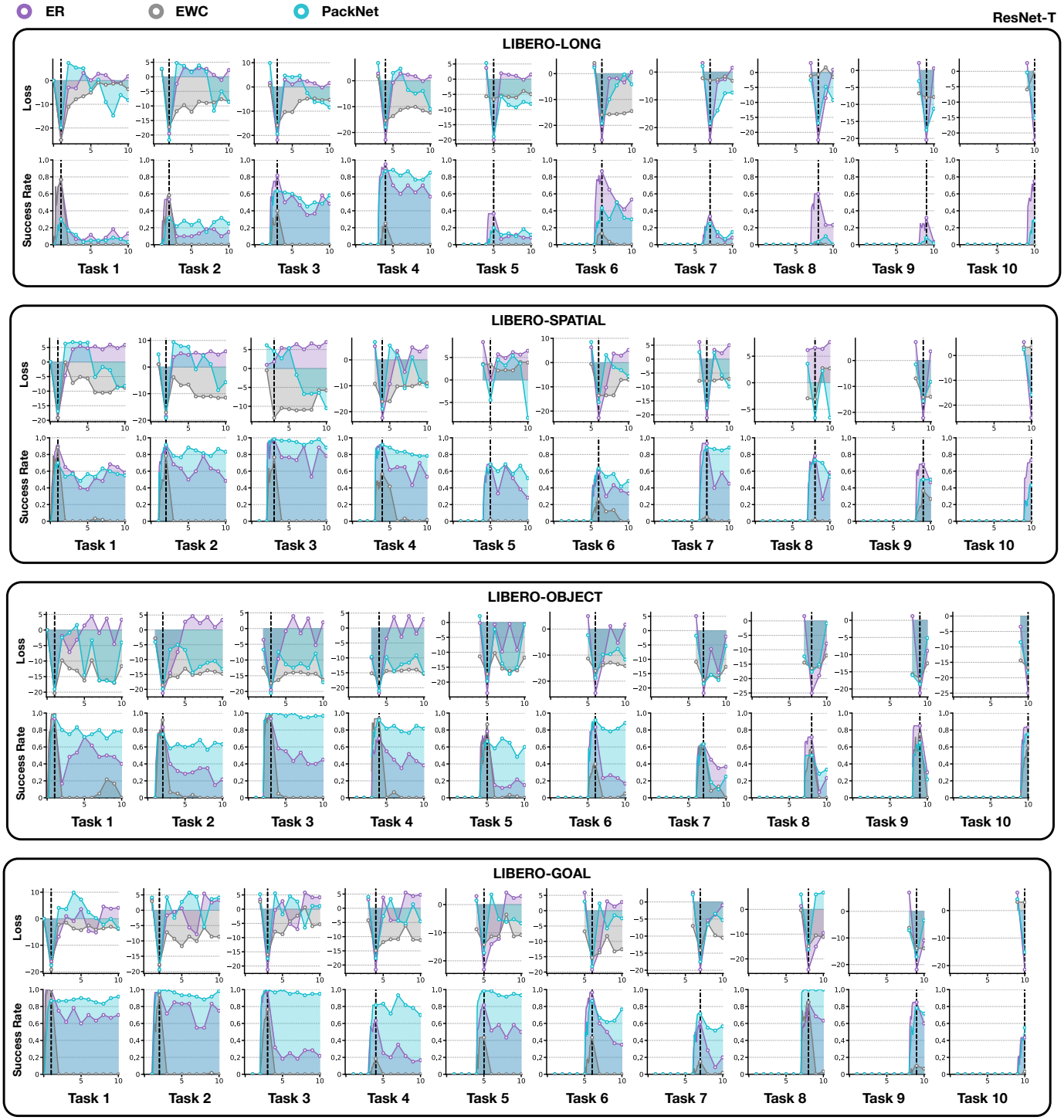


Fig. 18. Losses and success rates of ER (violet), EWC (grey), and PackNet (blue) on four task suites with ResNet-T policy. The first (second) row shows the loss (success rate) of the agent on task i throughout the LLDM procedure.

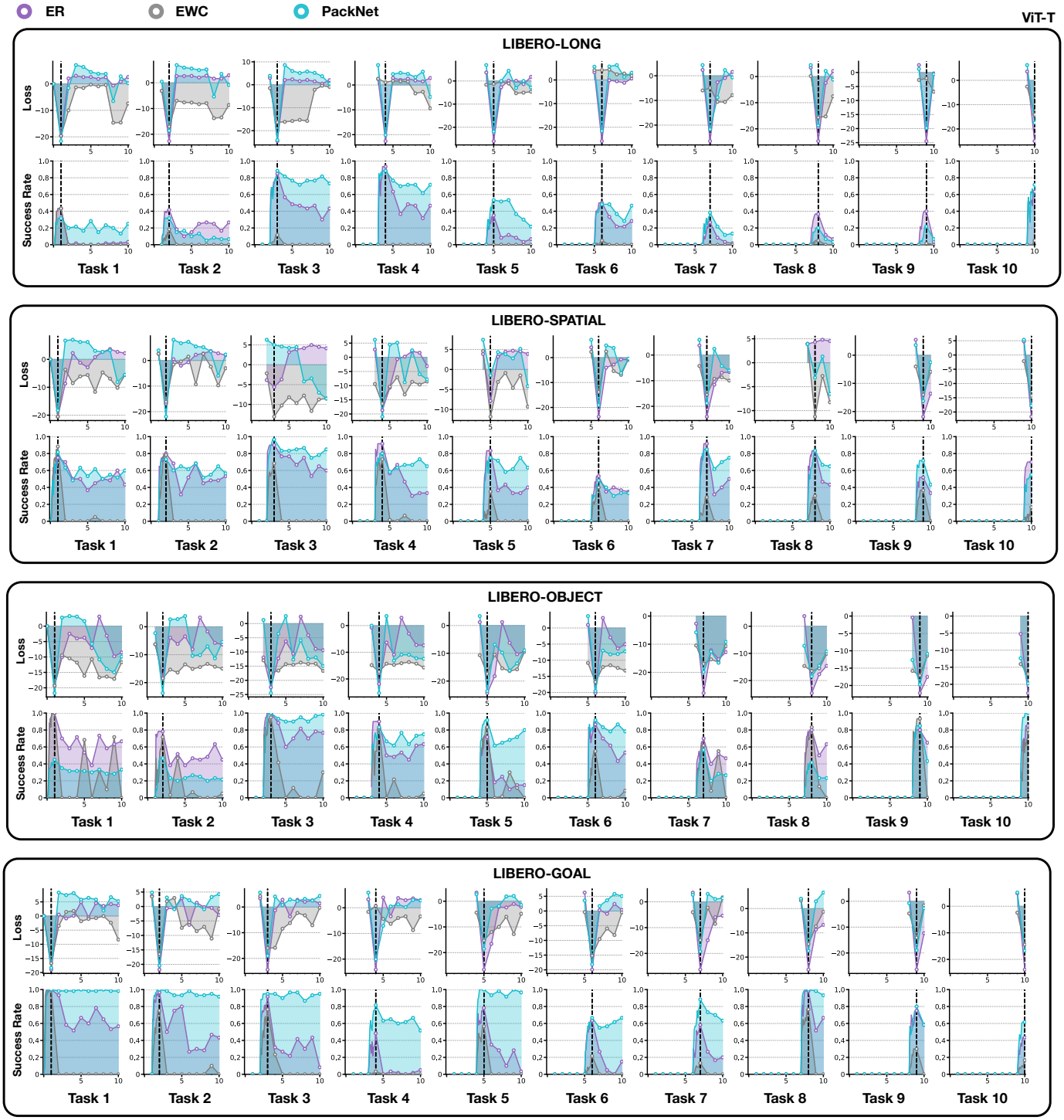


Fig. 19. Losses and success rates of ER (violet), EWC (grey), and PACKNET (blue) on four task suites with VIT-T policy. The first (second) row shows the loss (success rate) of the agent on task i throughout the LLDM procedure.

G. Attention Visualization

It is also important to visualize the behavior of the robot and its attention maps during the completion of tasks in the lifelong learning process to give us intuition and qualitative feedback on the performance of different algorithms and architectures. We visualize the attention maps of learned policies with greydanus2018visualizing and compare them in different studies as in D-C to see if the robot correctly pays attention to the right regions of interest in each task.

a) Perturbation-based attention visualization:: We use a perturbation-based method [57] to extract attention maps from agents. Given an input image I, the method applies a Gaussian filter to a pixel location (i, j) to blur the image partially, and produces the perturbed image $\Phi(I, i, j)$. Denote the learned policy as π and the inputs to the spatial module (e.g., the last latent representation of resnet or ViT encoder) $\pi_u(I)$ for image I. Then we define the saliency score as the Euclidean distance between the latent representations of the original and the blurred images:

$$S_\pi(i, j) = \frac{1}{2}\left|\left|\pi_u(I) - \pi_u(\Phi(I, i, j))\right|\right|^2. \tag{2}$$

Intuitively, $S_\pi(i, j)$ describes *how much removing information from the region around location* (i, j) *changes the policy*. In other words, a large $S_\pi(i, j)$ indicates that the information around pixel (i, j) is important for the learning agent's decision-making. Instead of calculating the score for every pixel, [57] found that computing a saliency score for pixel i mod 5 and j mod 5 produced good saliency maps at lower computational costs for Atari games. The final saliency map P is normalized as $P(i, j) = \frac{S_\pi(i,j)}{\sum_{i,j} S_\pi(i,j)}$.

We provide the visualization and our analysis on the following pages.

Different Task Suites

Findings: Figure 20 shows attention visualization for 12 tasks across 4 task suites (e.g., 3 tasks per suite). We observe that:

1) policies pay more attention to the robot arm and the target placement area than the target object.
2) sometimes the policy pays attention to task-irrelevant areas, such as the blank area on the table.

These observations demonstrate that the learned policy use perceptual data for decision-making in a very different way from how humans do. The robot policies tends to spuriously correlate task-irrelevant features with actions, a major reason why the policies overfit to the tasks and do not generalize well across tasks.

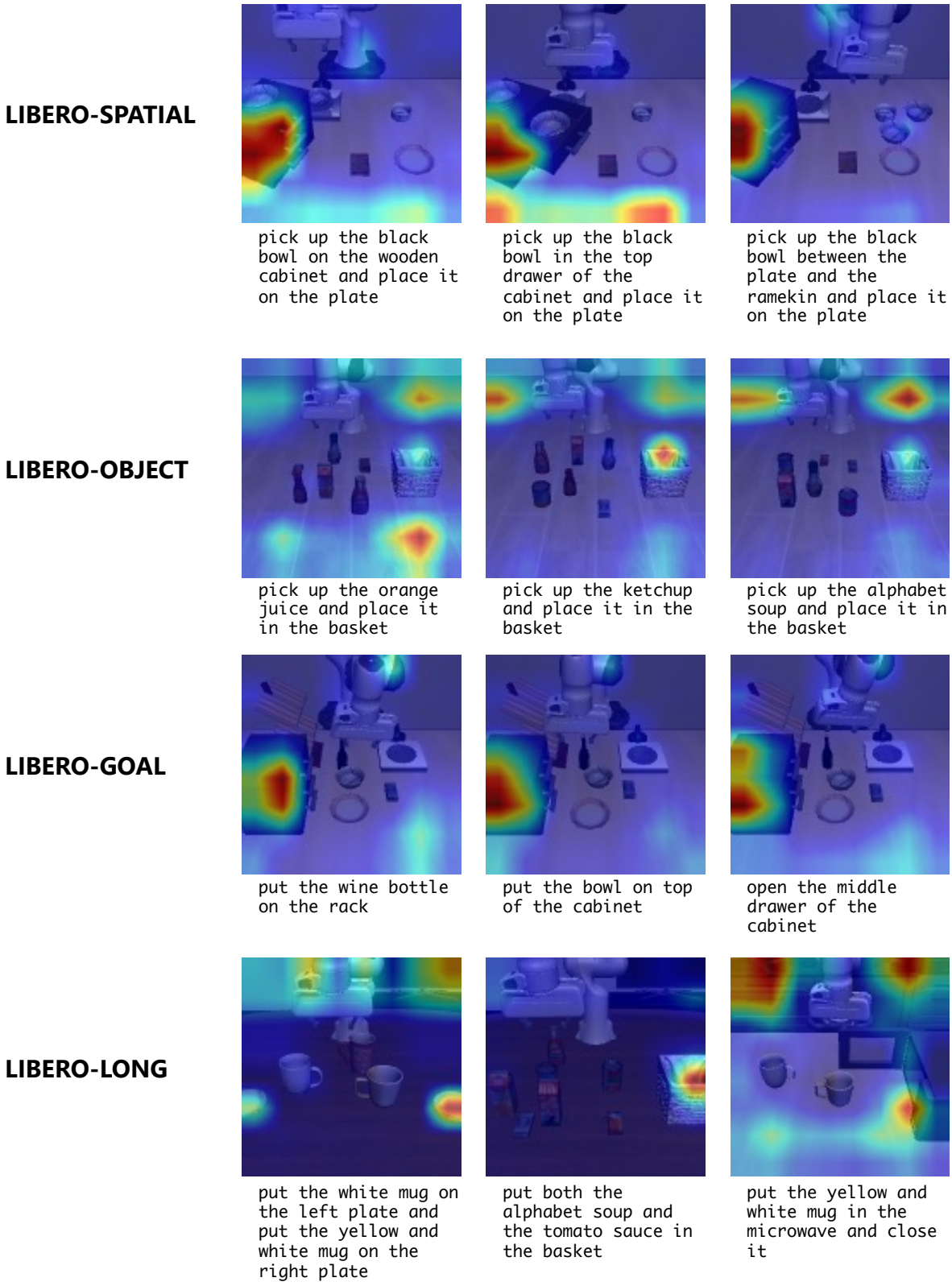


Fig. 20. Attention map comparison among different task suites with ER and ResNet-T. Each row corresponds to a task suite.

The Same Task over the Course of Lifelong Learning

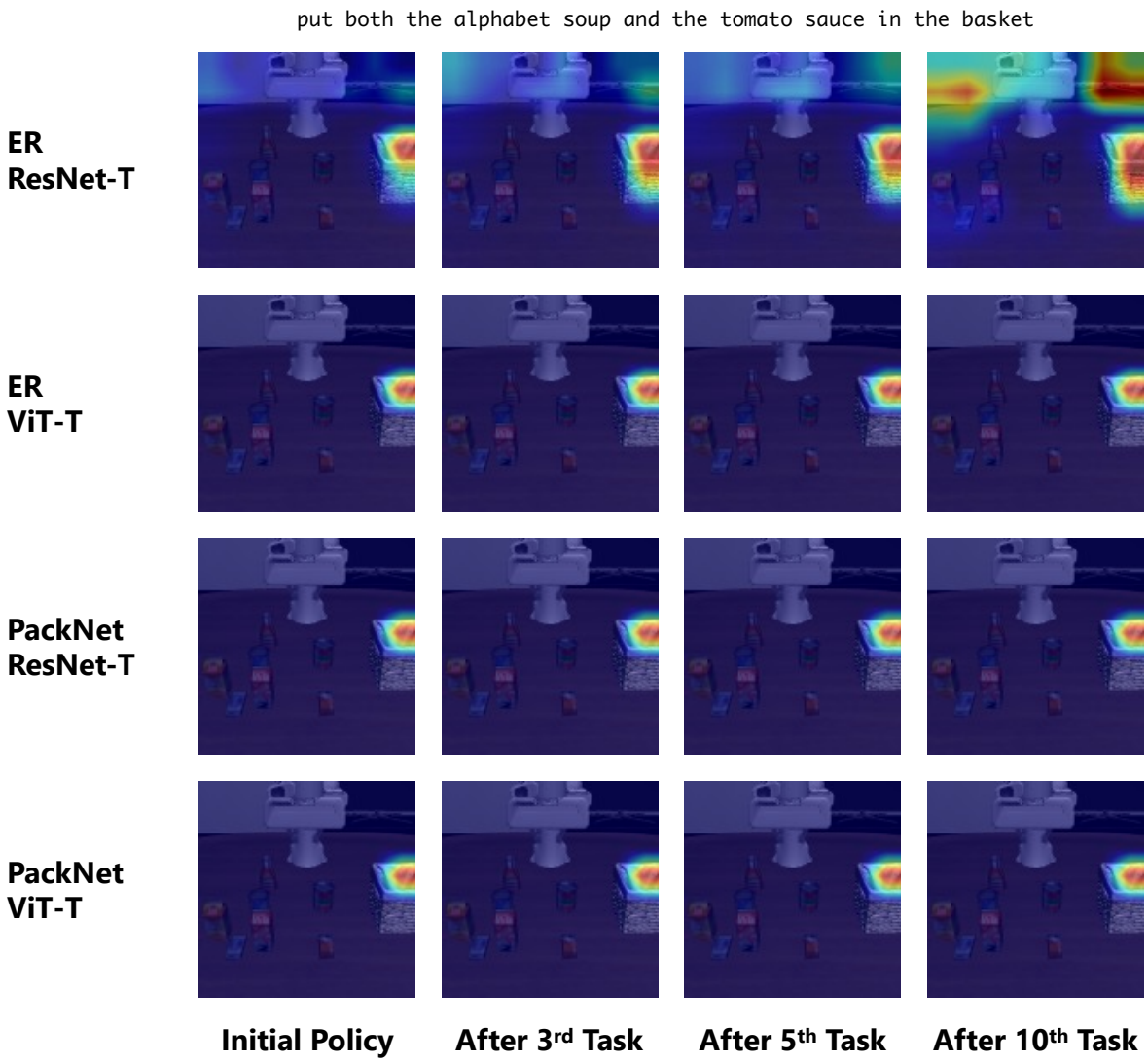


Fig. 21. Attention map of the same state of the task *put both the alphabet soup and the tomato sauce in the basket* from LIBERO-LONG during lifelong learning. Each row visualizes how the attention maps change on the first task with one of the LL algorithms (ER and PACKNET) and one of the neural architectures (RESNET-T and VIT-T). Initial policy is the policy that is trained on the first task. And all the following attention maps correspond to policies after training on the third, fifth, and the tenth tasks.

Findings: Figure 21 shows attention visualizations from policies trained with ER and PACKNET using the architectures RESNET-T and VIT-T respectively. We observe that:

1) The ViT visual encoder's attention is more consistent over time, while the ResNet encoder's attention map gradually dilutes.
2) PackNet, as it splits the model capacity for different tasks, shows a more consistent attention map over the course of learning.

Different Lifelong Learning Algorithms

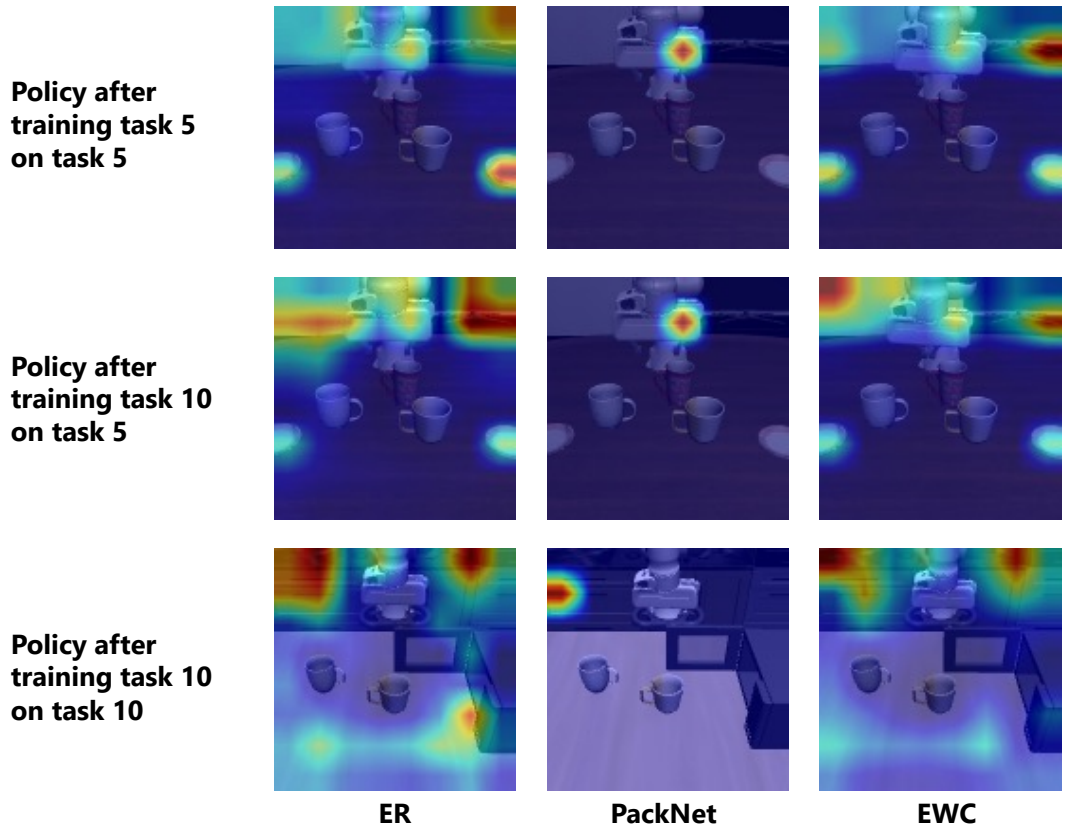


Fig. 22. Comparison of attention maps of different lifelong learning algorithms with RESNET-T on LIBERO-LONG. Each row shows the same state of a task with different neural architectures. "Task 5" refers to the task *put the white mug on the left plate and put the yellow and white mug on the right plate*. "Task 10" refers to the task *put the yellow and white mug in the microwave and close it*. The second row shows the policy that is trained on task 10 and gets evaluated on task 5, showing the attention map differences in backward transfer.

Findings: Figure 22 shows the attention visualization of three lifelong learning algorithms on LIBERO-LONG with RESNET-T on two tasks (task 5 and task 10). The first and third rows show the attention of the policy on the same task it has just learned. While the second row shows the attention of the policy on the task it learned in the past. We observe that:

1) PACKNET shows more concentrated attention compared against ER and EWC (usually just a single mode).
2) ER shares similar attention map with EWC, but EWC performs much worse than ER. Therefore, attention can only assist the analysis but cannot be treated as a criterion for performance prediction.

Different Neural Architectures

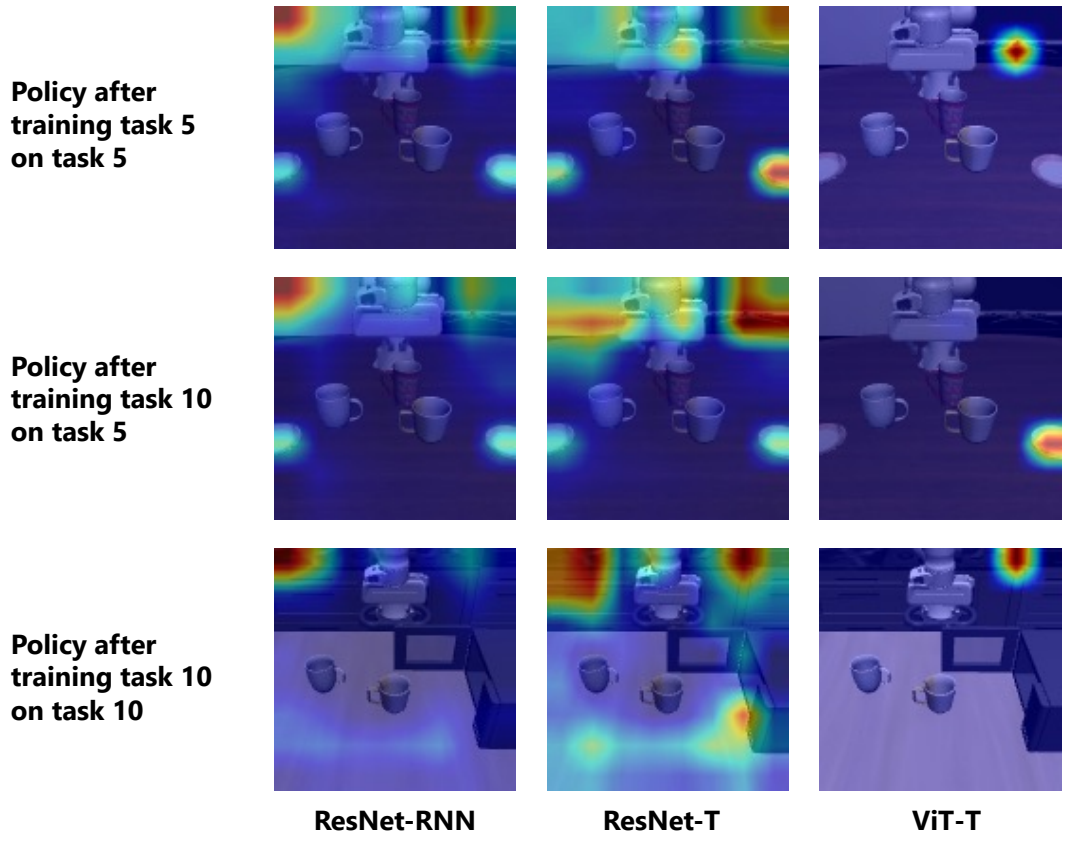


Fig. 23. Comparison of attention maps of different neural architectures with ER on LIBERO-LONG. Each row shows the same state of a task with different neural architectures. "Task 5" refers to the task *put the white mug on the left plate and put the yellow and white mug on the right plate*. "Task 10" refers to the task *put the yellow and white mug in the microwave and close it*. The second row shows the policy that is trained on task 10 and gets evaluated on task 5, showing the attention map differences in backward transfer.

Findings: Figure 23 shows attention map comparisons of the three neural architectures on LIBERO-LONG with ER on two tasks (task 5 and task 10). We observe that:

1) ViT has more concentrated attention than policies using ResNet.
2) When ResNet forgets, the attention is changing smoothly (more diluted). But for ViT, when it forgets, the attention can completely shift to a different location.
3) When ResNet is combined with LSTM or a temporal transformer, the attention hints at the "course of future trajectory". But we do not observe that when ViT is used as the encoder.

Different Task Ordering

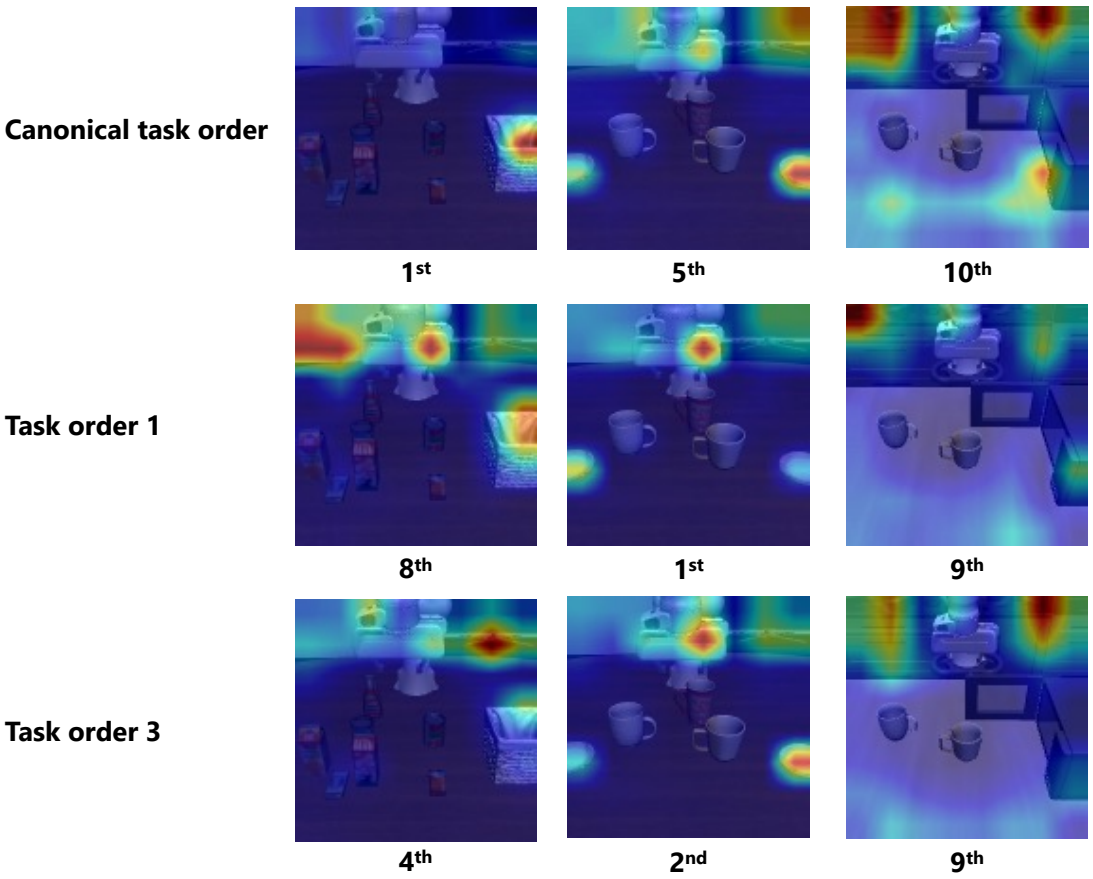


Fig. 24. Attention map comparison among different orderings with ER and RESNET-T on three selected tasks from LIBERO-LONG: *put both the alphabet soup and the tomato sauce in the basket*, *put the white mug on the left plate and put the yellow and white mug on the right plate*, and *put the yellow and white mug in the microwave and close it*. Each row corresponds to a specific sequence of task ordering, and the caption of each attention map indicates the order of the task in that sequence.

Findings: Figure 24 shows attention map comparisons of three different task orderings. We show two immediately learned tasks from LIBERO-LONG trained with ER and RESNET-T. We observe that:

1) As expected, learning the same task at different positions in the task stream results in different attention visualization.
2) There seems to be a trend that the policy has a more spread-out attention when it learns on tasks that are later in the sequence.

With or Without Pretraining

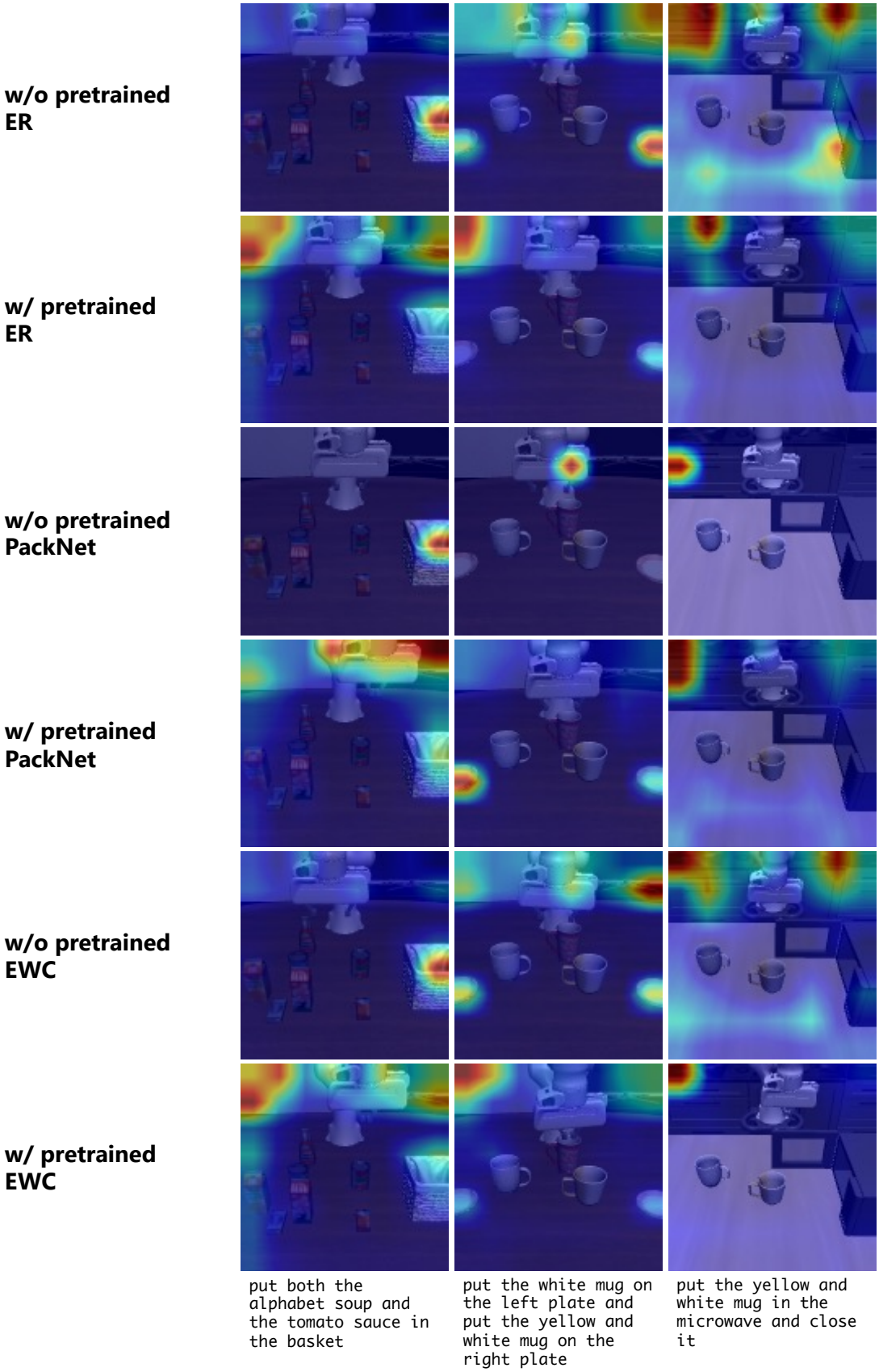


Fig. 25. Attention map comparison between models without/with pretrained models using RESNET-T and different lifelong learning algorithms on three selected tasks from LIBERO-LONG.

Findings: Figure 25 shows attention map comparisons between models with/without pretrained models on LIBERO-LONG with RESNET-T and all three LL algorithms. We observe that:

1) With pretraining, the policies attend to task-irrelevant regions more easily than those without pretraining.
2) Some of the policies with pretraining have better attention to the task-relevant features than their counterparts without pertaining, but their performance remains lower (the last in the second row and the second in the fourth row). This observation, again, shows that there is no positive correlation between semantically meaningful attention maps and the policy's performance.