A Metrics Visualization

We provide a visualization of the three metrics we compute in Figure 4. For completeness, we also provide the formulas for the metrics here:

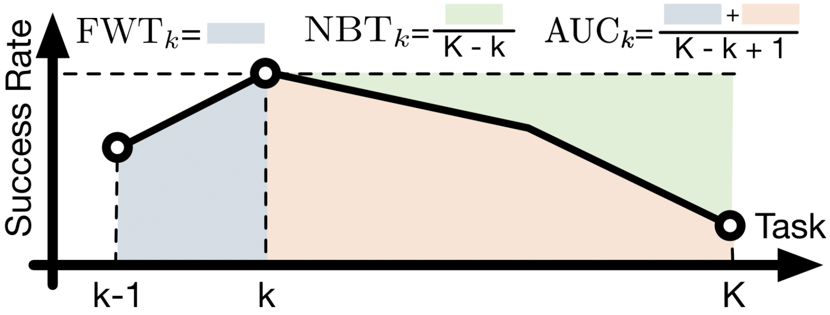


Figure 4: Metrics for LLDM

$$
\begin{aligned}
\text{FWT} &= \sum_{k\in[K]} \frac{\text{FWT}_k}{K}, \quad \text{FWT}_k = \frac{1}{11}\sum_{e\in\{0\dots 50\}} c_{k,k,e} \\
\text{NBT} &= \sum_{k\in[K]} \frac{\text{NBT}_k}{K}, \quad \text{NBT}_k = \frac{1}{K-k}\sum_{\tau=k+1}^{K} \big(c_{k,k} - c_{\tau,k}\big) \\
\text{AUC} &= \sum_{k\in[K]} \frac{\text{AUC}_k}{K}, \quad \text{AUC}_k = \frac{1}{K-k+1}\big(\text{FWT}_k + \sum_{\tau=k+1}^{K} c_{\tau,k}\big).
\end{aligned}
$$

B Implemented Neural Architectures and Lifelong Learning Algorithms

Neural Policy Arch.	ResNet-RNN ResNet-T ViT-T
Lifelong Learning Algo.	SeqL EWC [33] ER [13] PackNet [40] MTL

Table 4: The implemented neural policy architectures and the lifelong learning algorithms in LIBERO.

B.1 Neural Architectures

In Section 4.4, 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 5. Furthermore, Table 5, 6, and 7 display the hyperparameters for the architectures used throughout all of our experiments.

C Computation

For all experiments, we use a single Nvidia A100 GPU or a single Nvidia A40 GPU (CUDA 11.7) with 8 16 CPUs for training and evaluation.

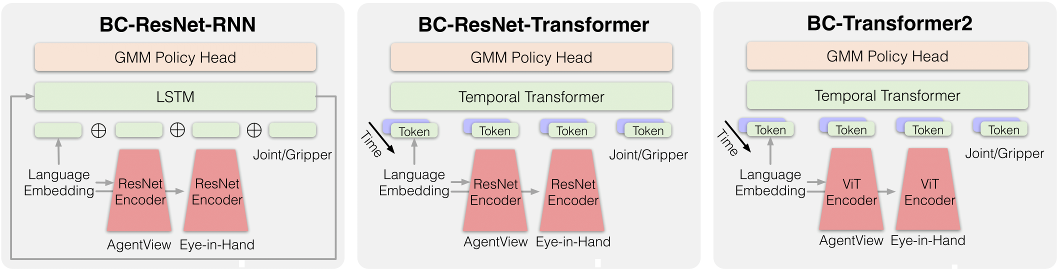


Figure 5: We provide visualizations of the architectures for RESNET-RNN, RESNET-T, and VIT-T, respectively. It is worth noting that each model architecture incorporates language embedding in distinct ways.

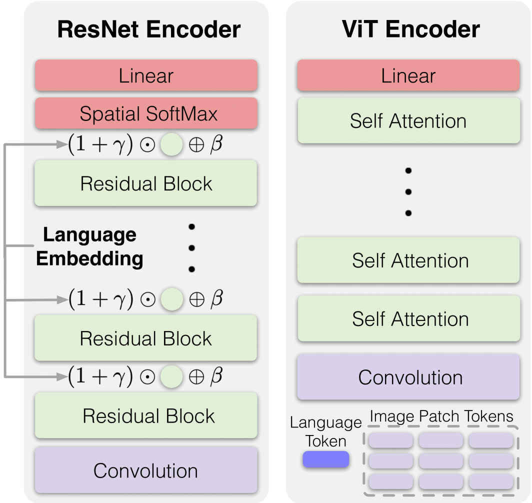


Figure 6: The image encoders: ResNet-based encoder and the vision transformer-based encoder.

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

Table 5: 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 6: Hyper parameters of RESNET-T.

C.1 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: Dynamic Architecture approaches, Regularization-Based approaches, and Rehearsal approaches. Although some recent works explore the combination of different approaches [2, 29, 53] or new strategies [72, 56, 14], our benchmark aims to provide an in-depth analysis of these three basic lifelong learning directions to reveal their pros and cons on robot learning tasks.

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 7: Hyper parameters of ViT-T.

The dynamic architecture approach gradually expands the learning model to incorporate new knowledge [55, 70, 40, 26, 69, 5]. Regularization-based methods, on the other hand, regularize the learner to a previous checkpoint when it learns a new task [33, 11, 59, 36]. Rehearsal methods save exemplar data from prior tasks and replay them with new data to consolidate the agent's memory [13, 38, 12, 9]. For a comprehensive review of LL methods, we refer readers to surveys [16, 48].

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

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.

EWC Elastic Weight Consolidation(EWC) [33] 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 $\lambda = 5 \cdot 10^4$.

PACKNET PACKNET [40] 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 [40] 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.

D LIBERO Task Suite Designs

D.1 Task Suites

We visualize all the tasks from the four task suites in Figure 7- 10. Figure 7 visualizes the initial states since the task goals are always the same. All the figures visualize the goal states of tasks except for Figure 7, which visualizes the initial states since the task goals are always the same.

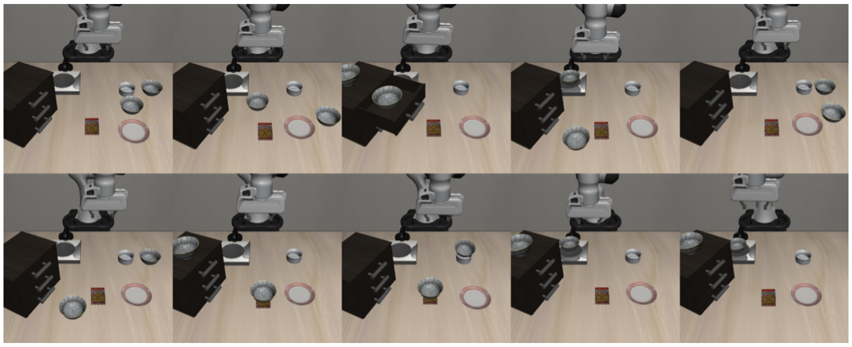


Figure 7: LIBERO-SPATIAL

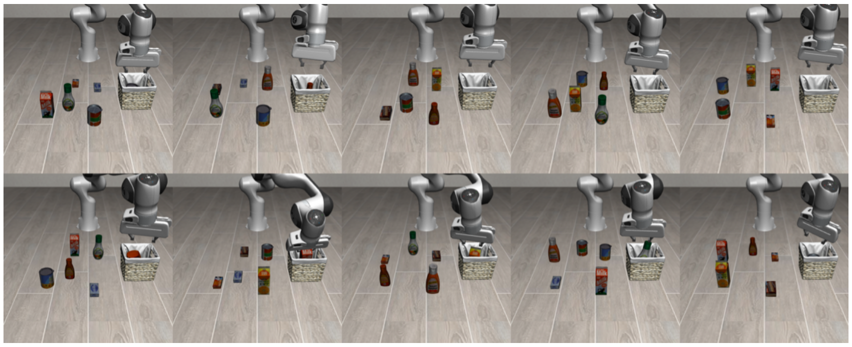


Figure 8: LIBERO-OBJECT

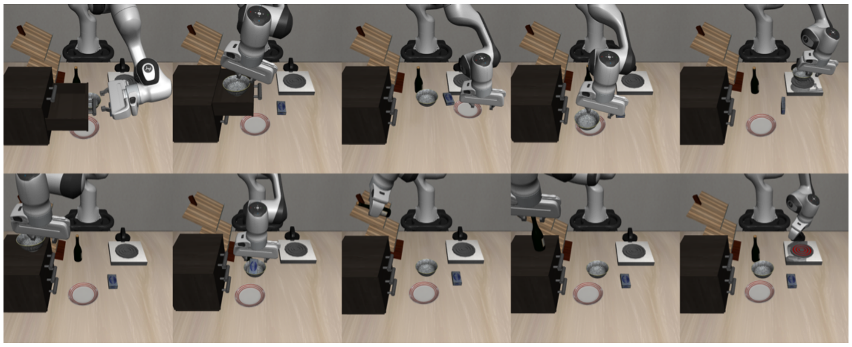


Figure 9: LIBERO-GOAL

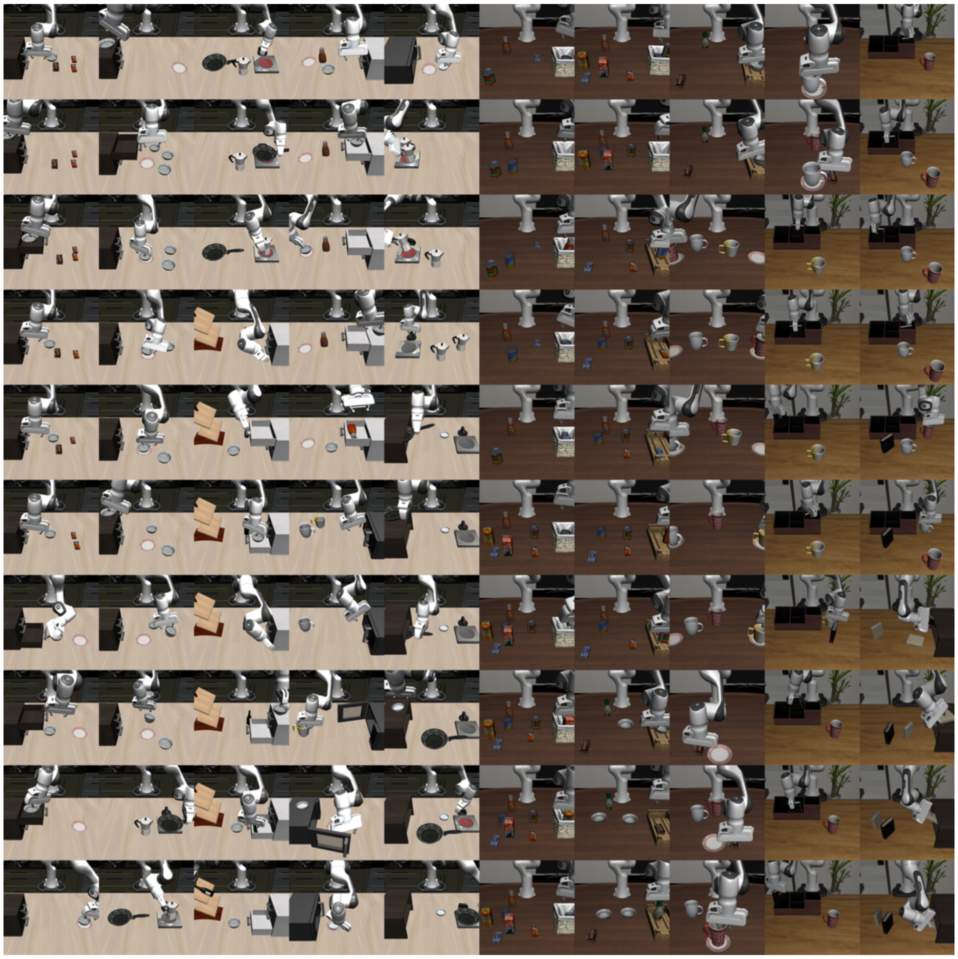


Figure 10: LIBERO-100

D.2 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 (
              (3.141592653589793 3.141592653589793)
            )
          )
      )
      (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)
    )
  )

)
```

E 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 [33], the replay-based method ER [13], and the dynamic architecture-based method PACKNET [40]. 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.1 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 [32] 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` [41], 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 $\sim$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 4 for the implemented algorithms and architectures.

F Additional Experiment Results

F.1 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 8.

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 11,12, and 13). Next, we provide plots that compare the performance of the agent using different policy architectures while fixing the lifelong learning algorithm (refer to Figure14, 15, and 16)

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 8: 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.

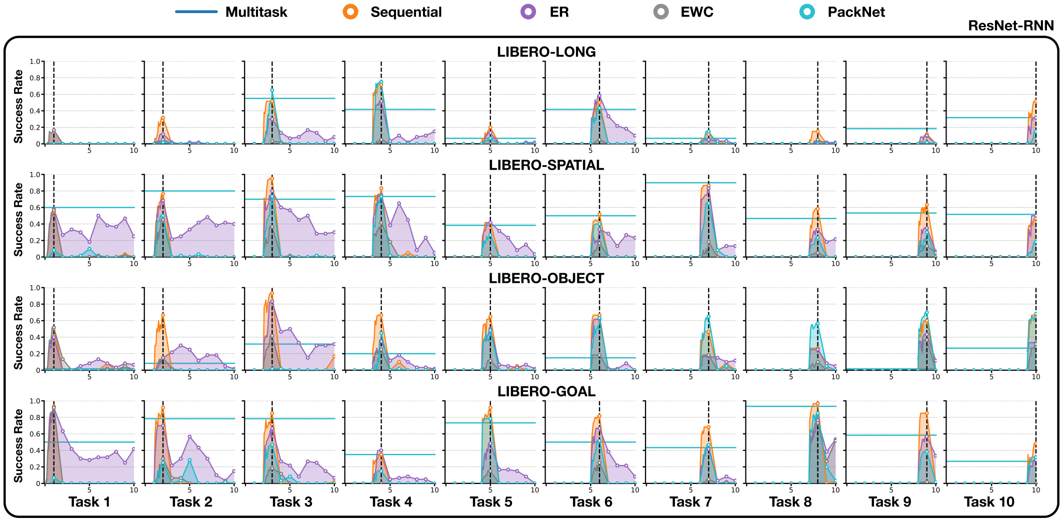


Figure 11: We compare the performance of different algorithms using the ResNet-RNN policy architecture in Figure 11. The y-axis represents the success rate, and the x-axis shows the agent's performance on each of the 10 tasks in a specific task suite over the course of learning. For example, the upper-left plot in the figure displays the agent's performance on the first task as it learns the 10 tasks sequentially.

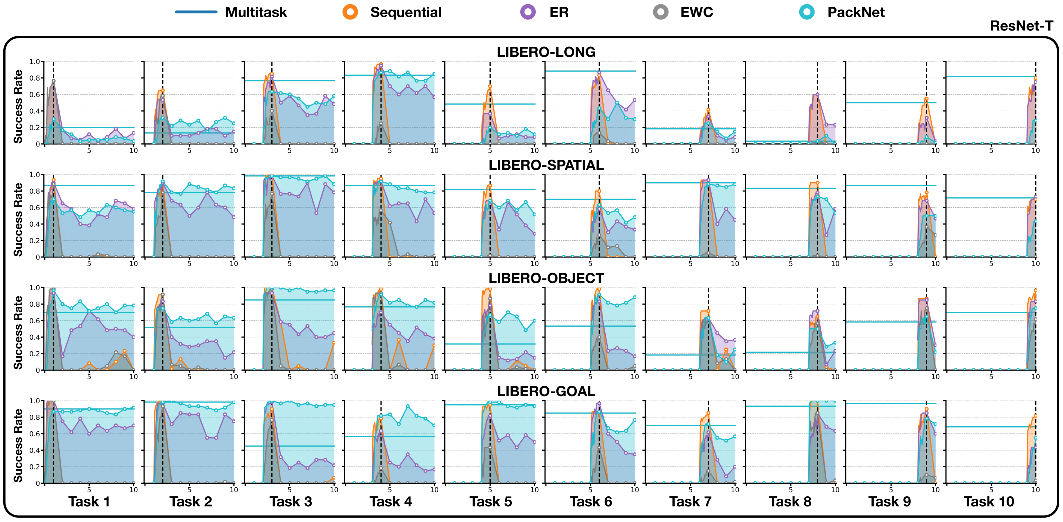


Figure 12: Comparison of different algorithms using the ResNet-T policy architecture. The y-axis represents the success rate, while the x-axis shows the agent's performance on each of the 10 tasks in a given task suite during the course of learning. For example, the plot in the upper-left corner depicts the agent's performance on the first task as it learns the 10 tasks sequentially.

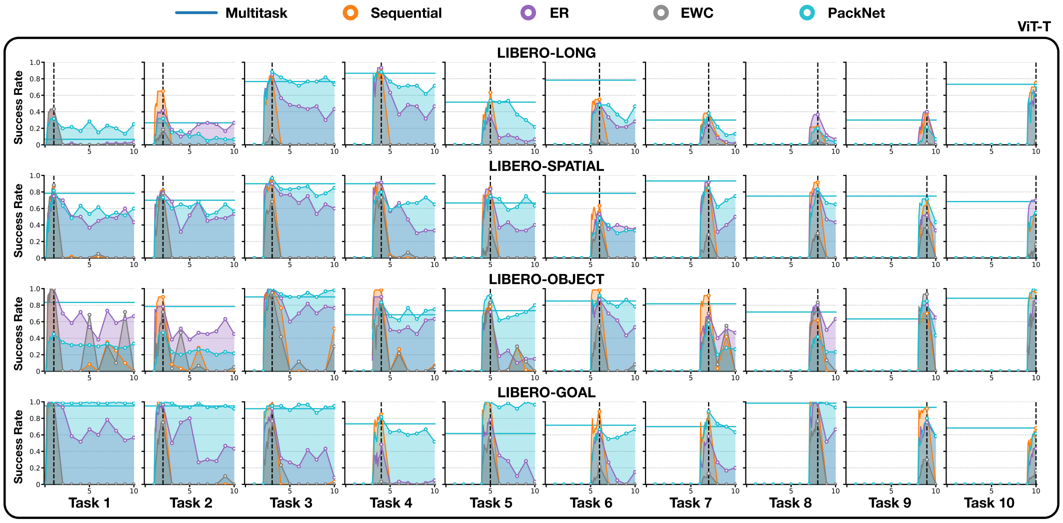


Figure 13: Comparison of different algorithms using the ViT-T policy architecture. The success rate is represented on the y-axis, while the x-axis shows the agent's performance on the 10 tasks in a given task suite over the course of learning. For instance, the plot in the upper-left corner illustrates the agent's performance on the first task when learning the 10 tasks sequentially.

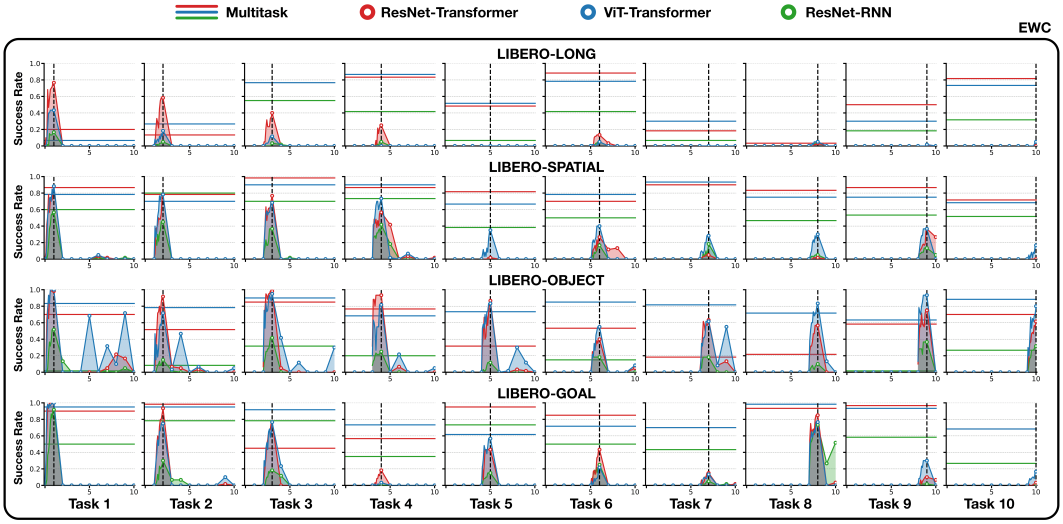


Figure 14: Comparison of different architectures with the EWC algorithm. The y-axis is the success rate, while the x-axis shows the agent's performance on the 10 tasks in a given task suite over the course of learning. For instance, the upper-left plot shows the agent's performance on the first task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

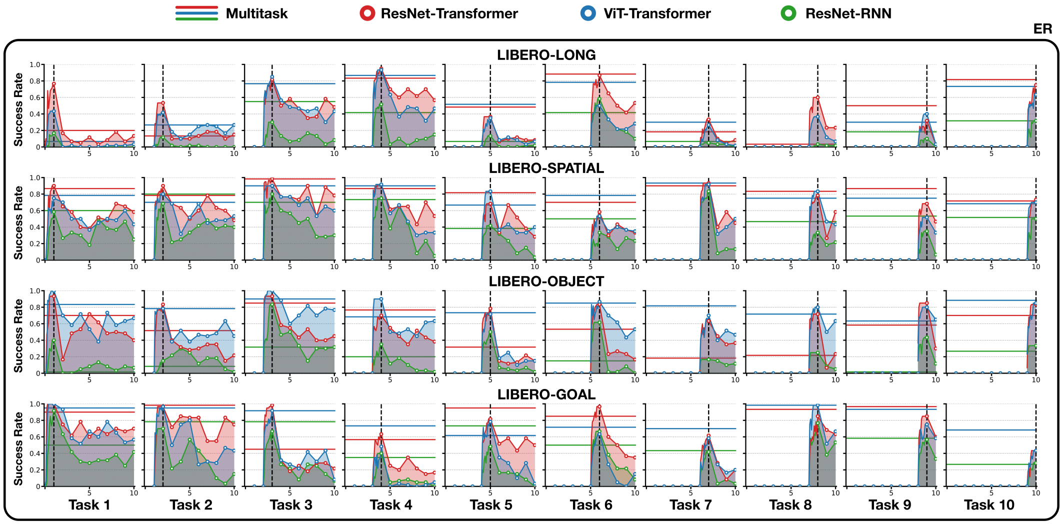


Figure 15: Comparison of different architectures with the ER algorithm. The y-axis is the success rate, while the x-axis shows the agent's performance on the 10 tasks in a given task suite ver the course of learning. For instance, the upper-left plot shows the agent's performance on the first task when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

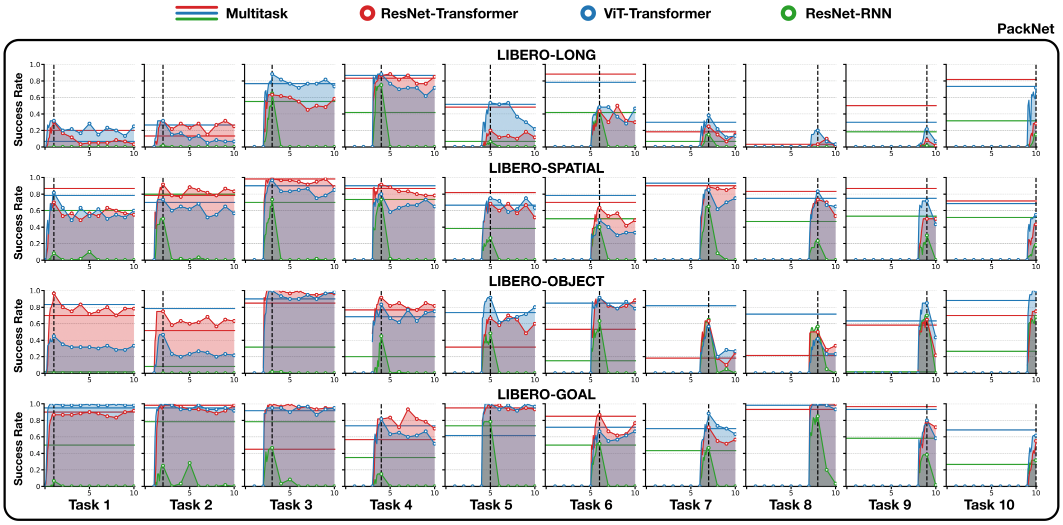


Figure 16: Comparison of different architectures with the PACKNET algorithm. The y-axis is the success rate, while the x-axis shows the agent's performance on the 10 tasks in a given task suite over the course of learning. For instance, the upper-left plot shows the agent's performance on the first task when learning the 10 tasks sequentially.

F.2 Study on task ordering (Q4)

Figure 17 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.

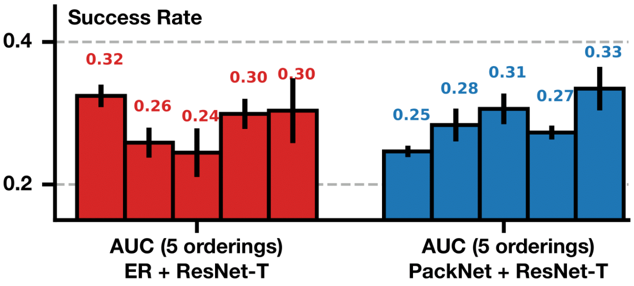


Figure 17: Performance of ER and PackNet using ResNet-T on five different task orderings. An error bar shows the performance standard deviation for a fixed ordering.

Findings: From Figure 17, 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.3 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 [54]. Figure 18, 12 and 13 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.

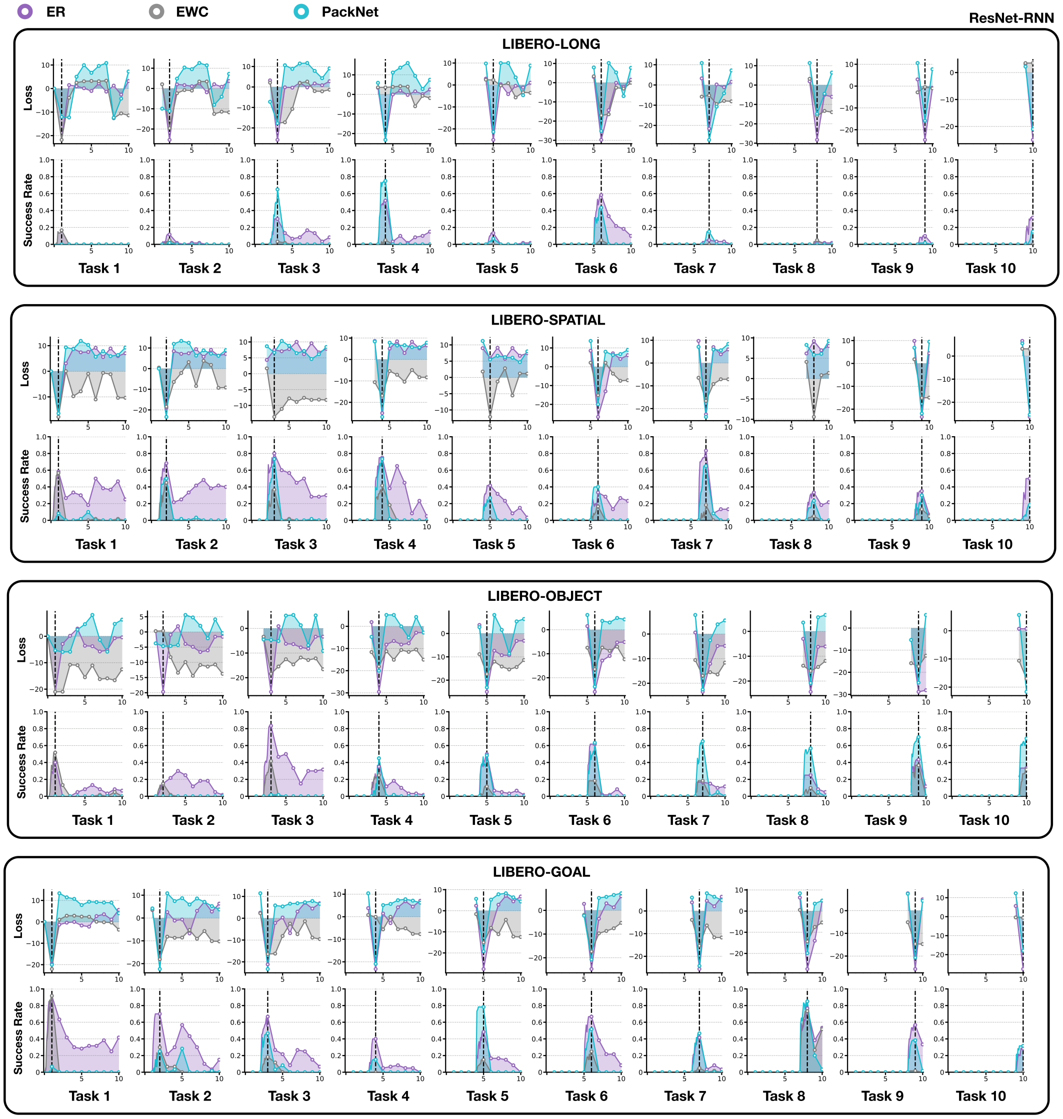


Figure 18: Losses and success rates of ER (violet), EWC (grey), and PackNet (blue) on four task suites with ResNet-RNN policy. The first (second) row shows the loss (success rate) of the agent on task i throughout the LLDM procedure.

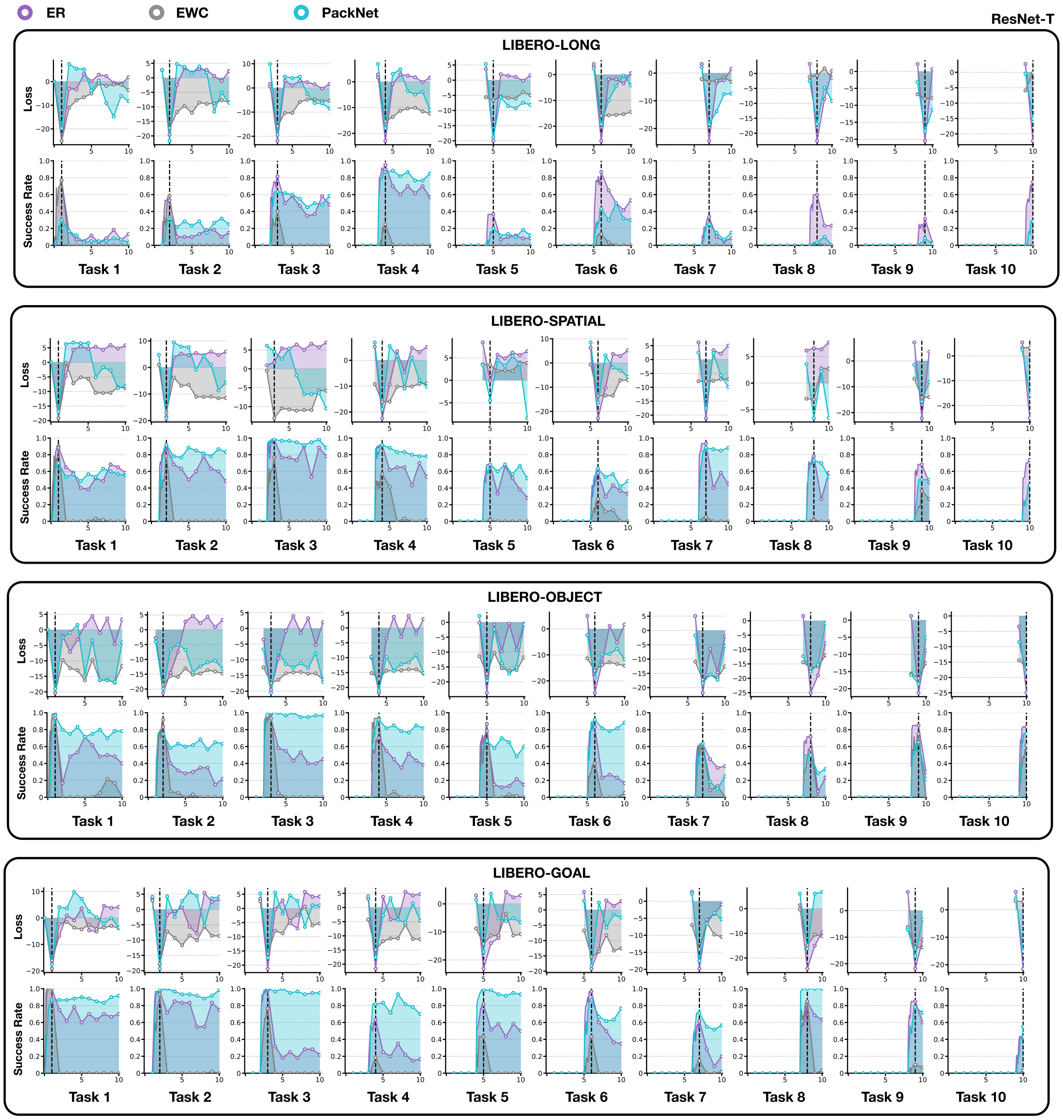


Figure 19: Losses and success rates of ER (violet), EWC (grey), and PACKNET (blue) on four task suites with RESNET-T policy. The first (second) row shows the loss (success rate) of the agent on task i throughout the LLDM procedure.

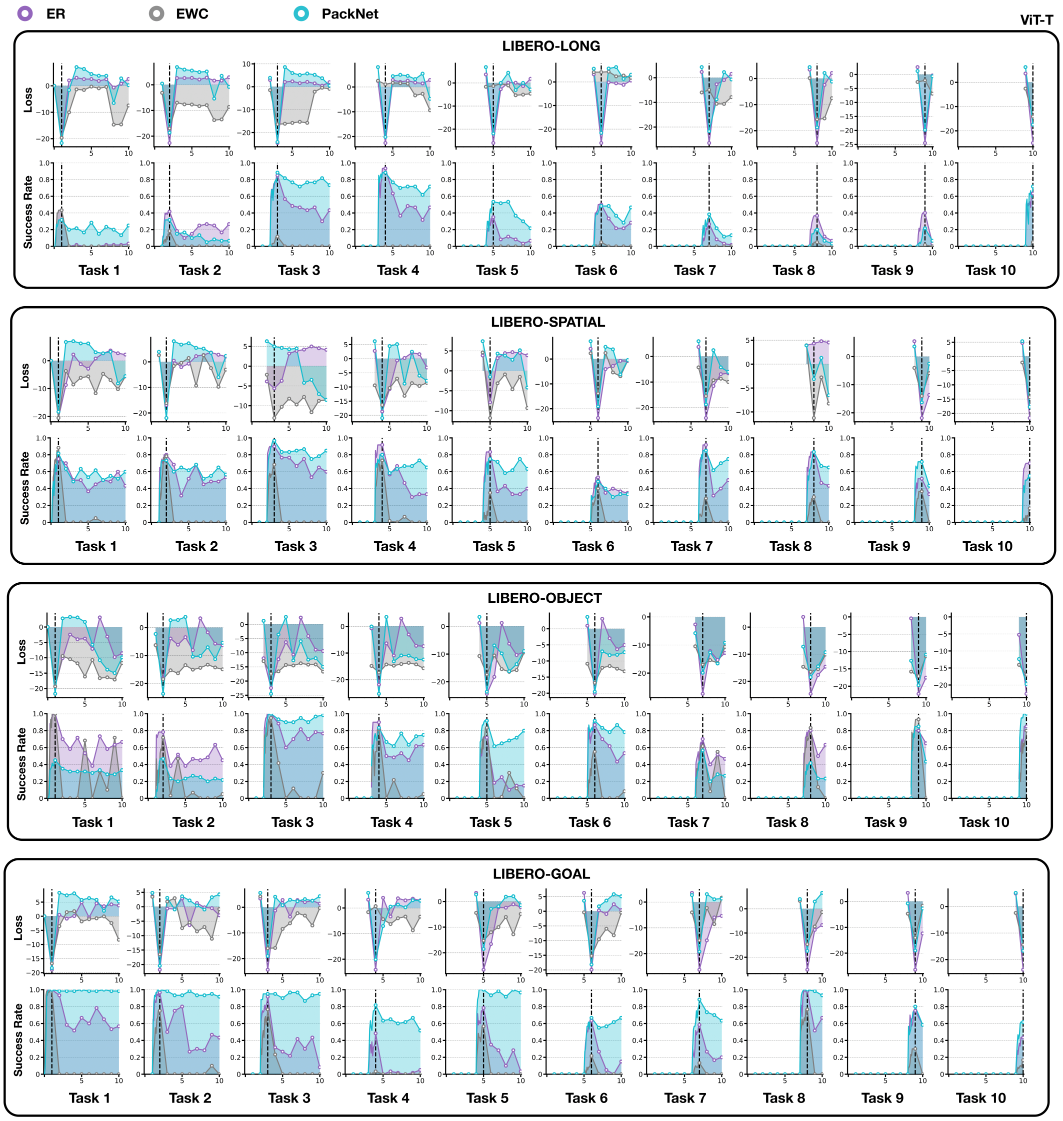


Figure 20: Losses and success rates of ER (violet), EWC (grey), and PackNet (blue) on four task suites with ViT-T policy. The first (second) row shows the loss (success rate) of the agent on task i throughout the LLDM procedure.

F.4 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 **(author?)** [23] and compare them in different studies as in 5.2 to see if the robot correctly pays attention to the right regions of interest in each task.

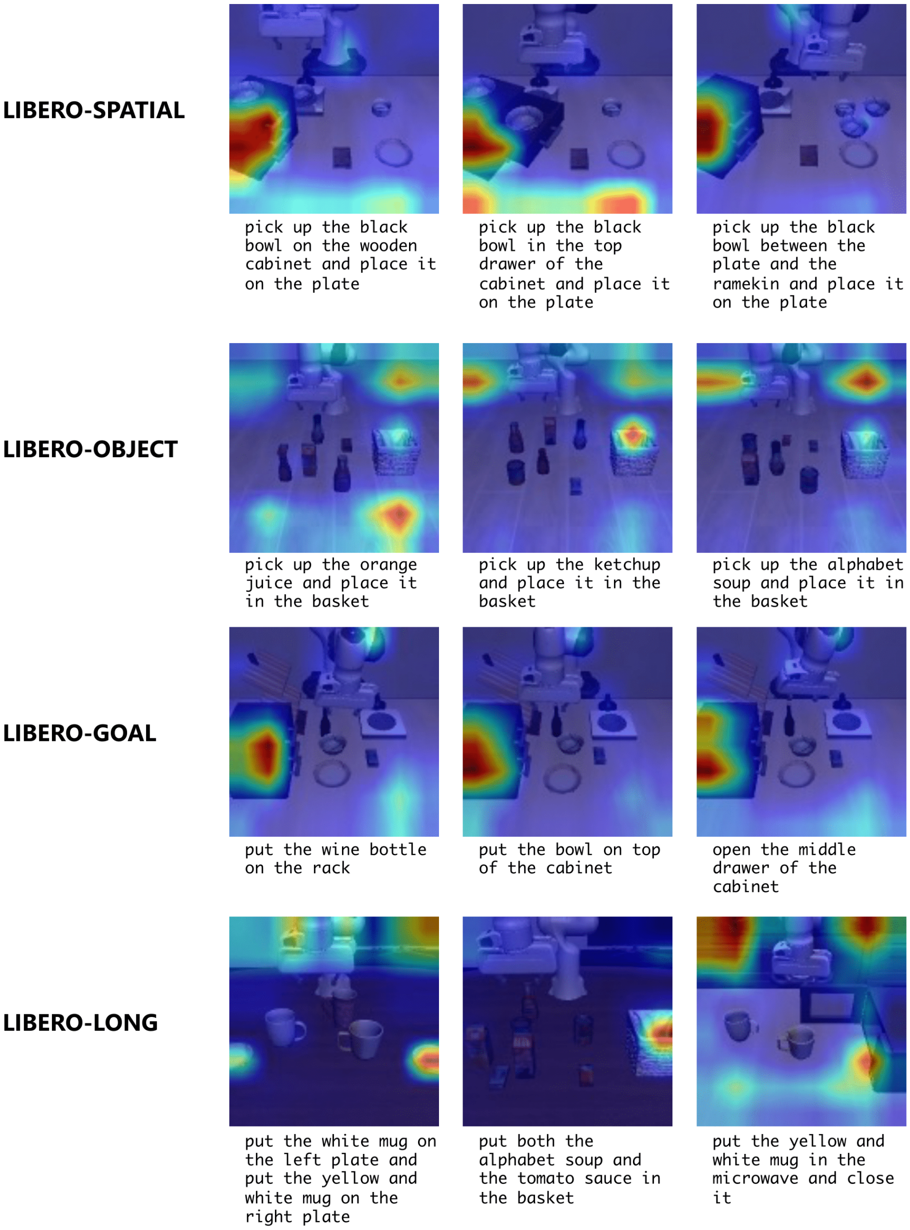
Perturbation-based attention visualization: We use a perturbation-based method [23] to extract attention maps from agents. Given an input image I, the method applies a Gaussian filter to a pixel location (i, j) to blur the image partially, and produces the perturbed image $\Phi(I, i, j)$. Denote the 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:

$$S_\pi(i, j) = \frac{1}{2}\left|\left|\pi_u(I) - \pi_u(\Phi(I, i, j))\right|\right|^2. \tag{3}$$

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, [23] 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



pick up the black bowl on the wooden cabinet and place it on the plate

pick up the black bowl in the top drawer of the cabinet and place it on the plate

pick up the black bowl between the plate and the ramekin and place it on the plate

pick up the orange juice and place it in the basket

pick up the ketchup and place it in the basket

pick up the alphabet soup and place it in the basket

put the wine bottle on the rack

put the bowl on top of the cabinet

open the middle drawer of the cabinet

put the white mug on the left plate and put the yellow and white mug on the right plate

put both the alphabet soup and the tomato sauce in the basket

put the yellow and white mug in the microwave and close it

Figure 21: Attention map comparison among different task suites with ER and ResNet-T. Each row corresponds to a task suite.

Findings: Figure 21 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.

The Same Task over the Course of Lifelong Learning

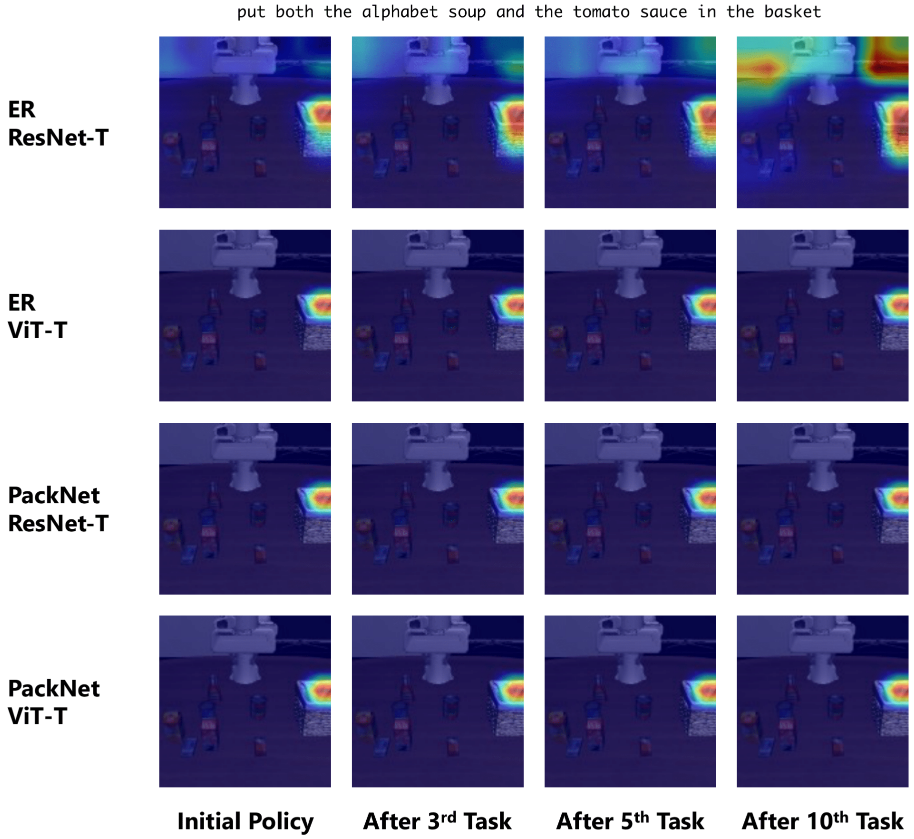


Figure 22: Attention map of the same state of the task *put both the alphabet soup and the tomato sauce in the basket* from LIBERO-LONG during lifelong learning. Each row visualizes how the attention maps change on the first task with one of the LL algorithms (ER and PACKNET) and one of the neural architectures (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 22 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

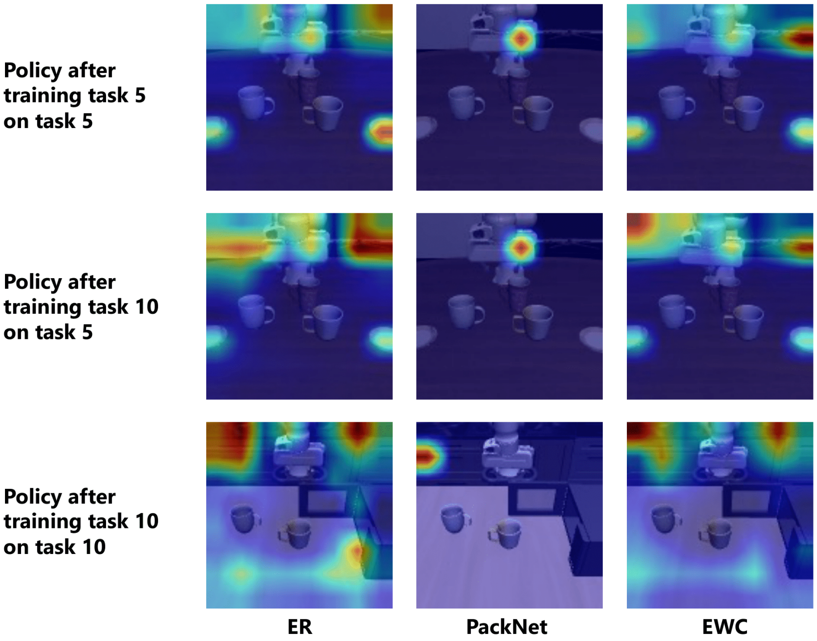

Policy after training task 5 on task 5
Policy after training task 10 on task 5
Policy after training task 10 on task 10
ER
PackNet
EWC

Figure 23: 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 23 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

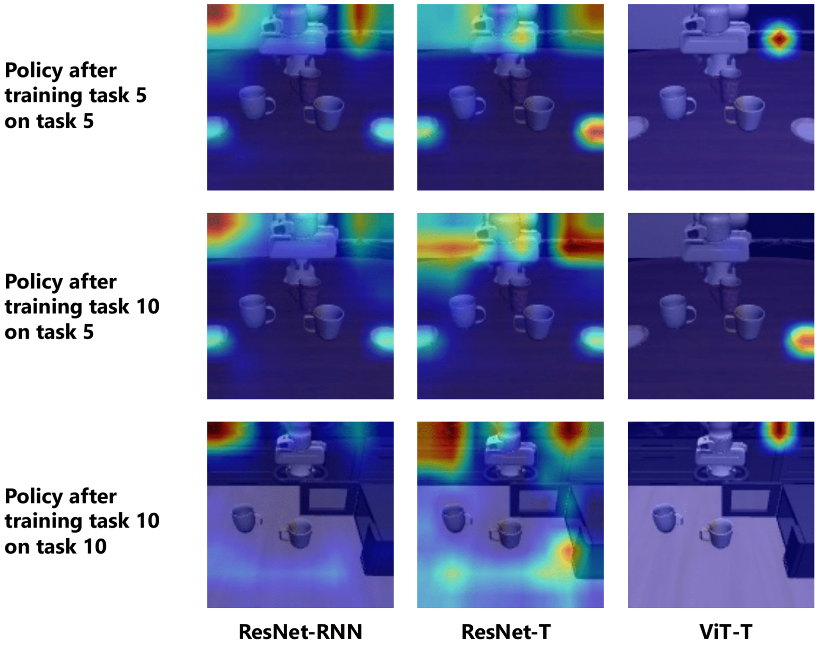

Policy after training task 5 on task 5
Policy after training task 10 on task 5
Policy after training task 10 on task 10
ResNet-RNN
ResNet-T
ViT-T

Figure 24: 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 24 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

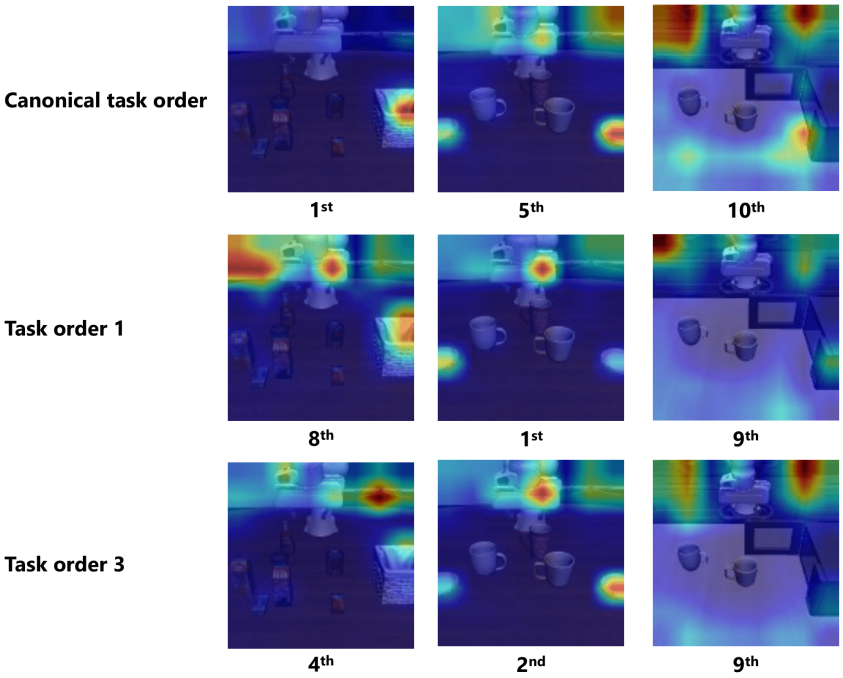


Figure 25: Attention map comparison among different orderings with ER and ResNet-T on three selected tasks from LIBERO-Long: *put both the alphabet soup and the tomato sauce in the basket*, *put the white mug on the left plate and put the yellow and white mug on the right plate*, and *put the yellow 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 25 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

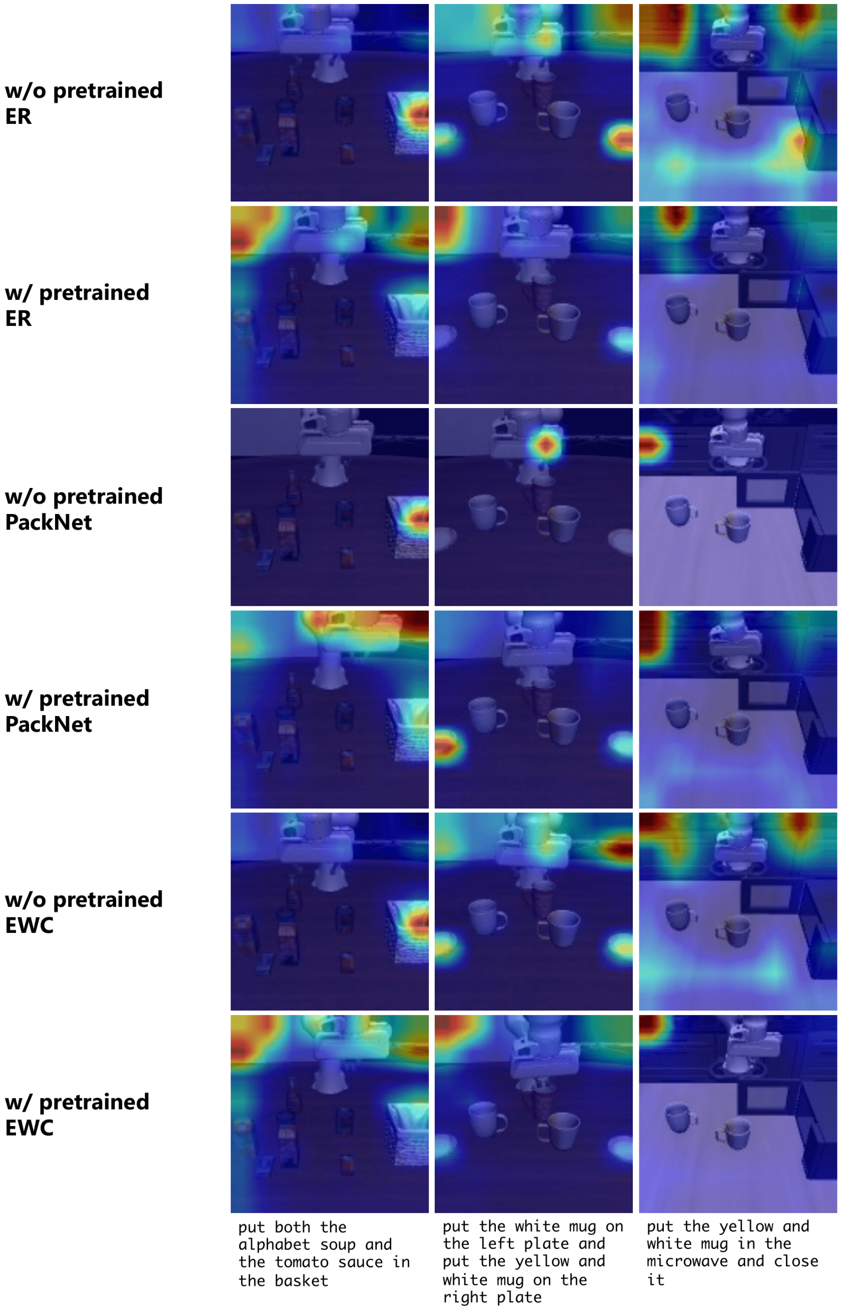


Figure 26: Attention map comparison between models without/with pretrained models using ResNet-T and different lifelong learning algorithms on three selected tasks from LIBERO-Long.

Findings: Figure 26 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.