Embodied Lifelong Learning for Task and Motion Planning

Anonymous Author(s)
Affiliation
Address
email

Abstract: A robot deployed in a home over long stretches of time faces a true lifelong learning problem. As it seeks to provide assistance to its users, the robot should leverage any accumulated experience to improve its own knowledge to become a more proficient assistant. We formulate this setting as a novel lifelong learning problem formulation in the context of learning for task and motion planning (TAMP). Exploiting the modularity of TAMP systems, we develop a generative mixture model that generates candidate continuous parameters for a planner. Whereas most existing lifelong learning approaches determine a priori how data is shared across task models, our approach learns shared and non-shared models and determines which to use online during planning based on auxiliary tasks that serve as a proxy for each model's understanding of a state. Our method exhibits substantial improvements in planning success on simulated 2D domains and on several problems from the BEHAVIOR benchmark [1].

Keywords: task and motion planning, lifelong learning, generative models

1 Introduction

Consider a home assistant robot operating over a lifetime in a home. The robot initially comes equipped with a number of basic capabilities for planning and control that enable it to execute certain actions, such as NAVIGATETO(`object`) and GRASP(`object`). The robot's user expects it to leverage these abilities to immediately assist with house chores, and to become increasingly capable over time, adapting to the types of problems that arise in its new home environment. This problem setting evokes a novel lifelong learning formulation for embodied intelligence that necessitates learning *in the field*, which we formalize and address in this work (Figure 1). Notably, we forgo any artificial separation between training and testing: the robot is continually asked to solve problems to the best of its abilities, and it is free to use any data it collects to improve its knowledge for future use.

One promising tool for tackling this lifelong learning challenge is planning. Planning models comprise relatively independent prediction modules, which permits learning disentangled knowledge. Similar forms of modularity have been shown to provide substantial leverage for training models continually by targeting the learning to individual modules and adding new modules over time [2].

In this paper, we focus on learning generative models to address the most difficult aspect of task and motion planning (TAMP) [3]: finding continuous parameters (grasps, poses, paths) that guarantee the success of a high-level plan. Our approach learns to generate samples that are conducive to problem completion, implicitly considering the effect of the current sample on future samples' success. Compared to other applications of generative models, one peculiarity of the lifelong sampler learning problem is that it is inherently multitask, since TAMP systems often apply similar basic skills (e.g., GRASP) to varied object types (e.g., `ball`, `box`). At two extremes of possible approaches to this multitask problem, we could learn an independent, specialized sampler for each object type, or we could learn a single, generic sampler across all types. Intuitively, specialized models are likely to

Submitted to the 7th Conference on Robot Learning (CoRL 2023). Do not distribute.

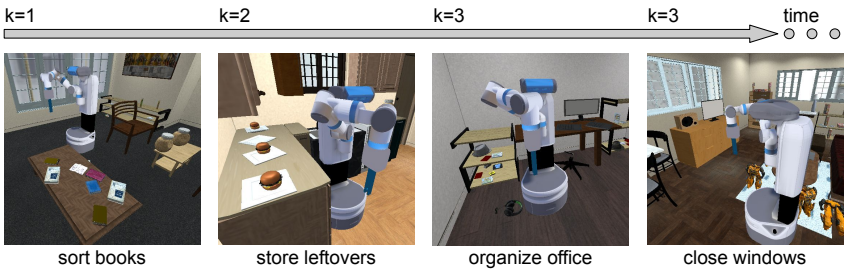


Figure 1: The learning robot will face a sequence of diverse TAMP problems in a true lifelong setting. It will use its current models to solve each problem as efficiently as possible, and then use any collected data to improve those models for the future. Images captured from BEHAVIOR [1].

yield better predictions, but they are trained on a smaller pool of data, making them potentially less robust. We propose a hybrid solution that learns both general and specialized samplers, and generates samples by determining online which model is most appropriate given a state. To do so, we pair each sampler with an auxiliary predictor. When drawing samples for solving a problem, the *measured* error on the auxiliary tasks serves as a proxy for the sampler's accuracy at the given state, which we use to construct a mixture distribution that assigns more weight to lower-error samplers.

We evaluate the resulting TAMP system for its cumulative performance as it encounters a sequence of diverse problems. This realistic formulation measures how effective the robot is, in aggregate, at solving problems over its entire lifetime. This contrasts with standard (artificial) lifelong learning evaluations, which instead focus on the final performance over the sequence of problems used for training. Concretely, we evaluate our approach on problems from a 2D domain and from the BEHAVIOR benchmark [1], demonstrating substantial improvements in planning performance.

2 The Lifelong Sampler Learning Problem

Our formulation of lifelong sampler learning emphasizes a realistic setting in which a robot is deployed in an environment and, from that moment on, is continually evaluated for its ability to solve TAMP problems. This section first formalizes the problem in terms of general TAMP systems, and next discusses how it applies specifically to search-then-sample (SeSamE) strategies.

2.1 Learning Over a Sequence of TAMP Problems

The lifelong learning robot faces a continual stream of planning problems $\tau^{(1)}, \tau^{(2)}, \ldots$. Each problem $\tau^{(k)} = \langle s^{(k)}_{\text{init}}, g^{(k)} \rangle \sim \mathcal{D}^{(k)}$ is defined by an initial state $s^{(k)}_{\text{init}}$ and a goal $g^{(k)}$, drawn sequentially from some problem distribution $\mathcal{D}^{(k)}$, which is potentially nonstationary (as indicated by the superscript k). We assume that the robot has access to a sound and probabilistically complete planner; while the planner is guaranteed to solve any problem given sufficient computation time, we expect many problems to be infeasible within a reasonable time. In consequence, the robot should leverage any available data to focus its planning process on choices that are likely to be successful.

Like in the real world, there are no distinct training and evaluation stages. Instead, at each time k, the robot seeks to construct a plan for a new problem $\tau^{(k)}$. We record whether the robot succeeds at the problem within a bounded number of samples B, and how many samples it attempts. The robot may use any experience it collects during the planning attempt as training data to improve its models. We measure performance as the cumulative number of problems solved within a given number of attempted samples. A system that learns efficiently from the data available so far will solve new problems more quickly, gaining even more data from successful plans, in a virtuous cycle.

One attribute of this formulation is that the agent is never evaluated on any previous problem—after time k, problem $\tau^{(k)}$ is never exactly encountered again. This departs from existing lifelong learning formulations, which evaluate the agent on previous problems $\tau^{(1)}, \ldots, \tau^{(k-1)}$ to measure forgetting.

Yet, avoiding forgetting still contributes to attaining good performance in our formulation. Since we evaluate the robot continually on *new* problems, it must generalize from prior experience to perform well. If the robot forgets past knowledge and overfits to the latest problems, then it will never grow stronger over the distribution of problems that it may face in the future. Even in nonstationary cases, where later problems differ substantially from earlier ones, retaining knowledge of past instances may improve generalization to future problems; we demonstrate this empirically in Section 5.3.

In order for the robot to generalize in this fashion, the problem distribution $\mathcal{D}^{(k)}$ must consist of problems that have some common underlying structure, enabling samplers to be shared across problems to serve as the medium through which prior experience informs future planning.

2.2 SeSamE Bilevel Planning

We formalize bilevel planning in a variant of PDDL [4] augmented with continuous parameters. The robot will be deployed in a world $\mathcal{W} = \langle \Theta, \mathcal{R}, \mathcal{S}, \mathcal{O}, \mathcal{A} \rangle$. Each object o of type $\theta_o \in \Theta$ is described by a feature vector $\boldsymbol{x}$. A state $s \in \mathcal{S}$ is characterized by the features $\boldsymbol{x}$ describing all objects. The conjunction of positive predicates $r \in \mathcal{R}$ on a state produces an abstract state $s^{\uparrow}$ (e.g., $\textsc{In}(\texttt{ball}, \texttt{box}) \wedge \textsc{On}(\texttt{box}, \texttt{floor})$). An abstract action $a^{\uparrow}$ is an operator from $\mathcal{O}$, with predicate-level preconditions and effects, augmented with a parameterized controller $\mathcal{C}^{a^{\uparrow}}_{\phi,o}$. The parameters of the controller are hybrid: o are discrete typed parameters that specify which objects to apply the controller to, while ϕ are continuous parameters that dictate how the action is applied. For example, a $\textsc{NavigateTo}$ action applied to $o = \texttt{table}$ could take continuous parameters $\phi = (\Delta_{\text{x}}, \Delta_{\text{y}})$ that specify the target coordinates, relative to the $\texttt{table}$. An action $a \in \mathcal{A}$ is the result of applying a controller with given parameters. We assume that all symbolic elements of $\mathcal{W}$ are given, and focus on determining the continuous parameters ϕ that result in actions that lead to complete plans. A planning problem $\tau = \langle s_{\text{init}}, g \rangle$ is given by a continuous-level initial state s_{init} and a predicate-level goal g. A satisficing plan π is a sequence of actions $a_1, \ldots, a_N$ which, when applied starting from s_{init}, results in an abstract state where $s^{\uparrow} \subseteq g$.

Algorithm 1 SeSamE(τ, N, M)

```
skeleton_gen ← discreteSearchSolutionGen(τ)
for j = 1, ..., N do
    skel ← skeleton_gen.next()
    for i = 1, ..., len(skel) do
        φ[i] ← sample(skel[i], s[i])
        s[i+1] ← simulate(s[i], skel[i], φ[i])
        if not valid(s[i+1]) then
            if sample_count < M then i--        ▷ Backtrack
            else break                          ▷ Try new skeleton
    if s[i] ∈ τ.goal then return skel, φ
```

The SeSamE strategy (Algorithm 1) first obtains a skeleton plan at the discrete level using standard planning techniques (e.g., A^*), and subsequently refines the skeleton into a low-level plan via sampling. While generating samples in the SeSamE setting is inexpensive, the simulation step that evaluates each sample can be quite expensive (e.g., due to inverse/forward kinematics, collision checking, or path planning). Reducing the number of samples required to obtain a satisficing plan, and hence the number of calls to the simulator, can substantially improve the efficiency of the TAMP system, vastly increasing the space of problems that can be solved in a tractable amount of time.

One challenge in this sampling-based formulation is specifying a distribution that simultaneously 1) covers the space of plausible solutions and 2) predominantly generates promising candidates. Learning such a distribution from data would constitute a major step toward constructing an effective TAMP system.

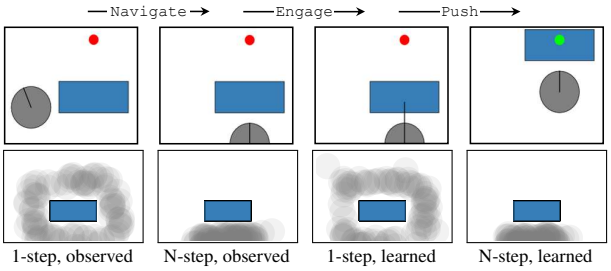


Figure 2: Visualization of sample distributions optimized for TAMP problem completion. Top: a 2D robot is tasked with pushing a block upward. Bottom: distribution of navigation parameters that achieve reachability (1-step) and that solve the overall problem (N-step), generated by a manually defined distribution (observed) and learned by a diffusion model (learned). The samples that consider global success yield a distribution over promising (as opposed to merely valid) actions. Diffusion models represent the distributions well, making it a promising representation for TAMP samplers.

2.3 Samplers as Generative Models of Plausible Candidate Action Parameters

The sampler for each abstract action $a^{\uparrow}$ should generate action parameters from a distribution conditioned on the current state: $\phi \sim p(\cdot \mid s, a^{\uparrow})$. This distribution should capture the whole space of action parameters that, given the current state, may lead to successful execution of an overall plan. To illustrate this point, consider the block-pushing domain in Figure 2 (see Section 5.1 for details). A naïve navigation sampler could bring the robot within reach of the block, but a more intelligent sampler would only consider navigating to the bottom of the block to enable pushing upward.

To learn a generative model that meets these criteria, we require access to a pool of paired states and action parameters (s, ϕ) that is representative of the choices that eventually (after successfully sampling all subsequent actions in a skeleton plan) lead to a satisficing plan. Critically, the action parameters that succeed for a given abstract action $p_{a^{\uparrow}}$ depend on the samplers for the remaining abstract actions, and as such the data must be collected jointly to ensure that the learned distributions are compatible with each other. The lifelong problem formulation of Section 2.1 satisfies this criterion.

3 Nested Models for Sparse Data

In our lifelong learning setting, the robot will have to use whatever data it has to make the best predictions possible. To handle the sparse-data setting that inevitably occurs early in the robot's experience, we formulate a solution that nests predictors of different generality.

In order to train neural generative models as the samplers for our TAMP system, we construct a vector representation of the conditioning variables. The sampling distribution for each abstract action $a^{\uparrow}$ is given by $p(\phi \mid s, \mathcal{C}^{a^{\uparrow}}_{\phi,o})$. We first represent the state s in terms of the continuous low-level features of the objects involved in the action, $\boldsymbol{x}_o$, following prior work [5]. A trivial second step is to train separate models for each form of controller, $p_{\mathcal{C}}$, since different controllers generally have different parameterizations, and so there are no commonalities across their sample distributions. However, each such controller may act on objects of different natures in ways that require substantially different parameters: this distinction is encoded in the discrete variable corresponding to the object types θ_o.

Thus, we would like the predictions of our model to be specialized to each individual object type. For this, we can consider the following two strategies (omitting the $\mathcal{C}$ subscript for clarity):

1. Learn a single model $p(\phi \mid \boldsymbol{x}_o, \theta_o)$ using, for example, a one-hot encoding of the types. This would enable the learning of a single, shared latent representation of $\boldsymbol{x}_o$ in the neural net model. However, in cases where the amount of data is small, it is more likely to simply overfit.
2. Learn a separate, simpler, model $p_\ell(\phi \mid \boldsymbol{x}_o)$ for each discrete value ℓ of θ_o.

However, early on, when the robot has observed very little data, we may prefer an even more aggressive strategy that pools data over all values of θ_o, and learns a single simple model $p(\phi \mid \boldsymbol{x}_o)$. This model will not be highly specific or accurate, but it may learn more quickly to generate reasonable suggestions, due to pooled data. It is also important to observe that, in the lifelong setting, we may have an asymmetric distribution of experience with samplers for different object types, so that some specific models would have substantially more training data available than others, in a way that cannot be predicted in advance and that will change over time.

For all these reasons, we propose a *nested* approach: train both a generic sampler $p(\phi \mid \boldsymbol{x}_o)$ and a collection of specialized samplers $p_\ell(\phi \mid \boldsymbol{x}_o)$ for all values ℓ of θ_o. Then, we decide *online*, given an input with $\theta_o = \ell$, based on an assessment of prediction reliability, how much to value predictions from the generic versus the specialized samplers, and actually sample from a mixture distribution.

To construct this assessment of the reliability of each generative model, we construct an auxiliary training task to predict a variable z, and augment each training example, so we have (s, ϕ, z). Critically, z must be a value that can be directly measured by the robot, so that the error between its learned predictor $f(s, \phi)$ and the actual observed z can serve as a measure of how well trained $p(\phi \mid s)$ is in the part of the input space near s. So, in parallel with training $p(\phi \mid s)$, we will train $f(s, \phi)$, and in parallel with training each $p_\ell(\phi \mid s)$, we will train a *separate* specialized $f_\ell(s, \phi)$.

At planning time, if the robot must draw a sample in state s for the action applied to objects with types $\ell = \theta_o$, it will use the following mixture distribution:

$$p_{\text{mix}}(\phi \mid s, z) = \frac{\rho(f(s,\phi), z)p(\phi \mid s) + \rho(f_\ell(s,\phi), z)p_\ell(\phi \mid s)}{\rho(f(s,\phi), z) + \rho(f_\ell(s,\phi), z)} \quad , \tag{1}$$

where $\rho(f_\ell(s,\phi), z)$ and $\rho(f(s,\phi), z) \in \mathbb{R}^+$ are *reliability measures* of the specialized and generic models at (s, ϕ), constructed by comparing their predictions to observed z (for example, the inverse of the squared prediction error). Note that, since the auxiliary signals z depend on the action parameters ϕ, the sampling process must draw samples from the two mixture components, use them to compute the reliability measures, and then choose between the samples based on the resulting mixture weights.

Implementation In our implementation, we use simple observable geometric properties of the world state as z values, restricted to the objects o that parameterize the action. Although they would not suffice for selecting good samples ϕ, since they do not consider the effect of ϕ on future steps nor the interactions with objects outside of o—like a complete simulator would—our ability to predict them accurately is a signifier that the accompanying model has obtained sufficient data in the neighborhood of (s, ϕ) to consider ϕ a good prediction. As an example, consider the NAVIGATETO(`block`) action from Figure 2. A useful set of auxiliary signals for this case may be: the orientation of the robot facing the `block`, the distance to the center of the `block`, the distance to the nearest point on the `block`'s boundaries, and the relative coordinates of this point and the robot's center in the `block`'s coordinate frame. Combined, these signals contain abundant information about the effects of the action, including those relevant for reachability and collision avoidance with the `block`. A model that learns to map a (s, ϕ) pair to accurate predictions of all these signals has likely observed sufficiently similar training pairs to generalize, and is therefore likely to generate high-quality samples ϕ. See Appendices A.2.2 and A.3.2 for the exact auxiliary signals used in our experiments.

4 Diffusion Models for Parameter Sampling

We use diffusion models to represent our learned samplers, due to their stability of training and their ability to model complex distributions. A diffusion model transforms Gaussian noise into a distribution over the sample space. To do so, it generates training data by following a forward diffusion process, which progressively adds Gaussian noise to observed training samples, and trains a reverse diffusion process that gradually denoises Gaussian noise to produce a sample from the learned distribution [6]. Concretely, when learning TAMP samplers, the forward diffusion process is given by $q(\phi_{0:T} \mid s) = q(\phi_0 \mid s) \prod_{t=1}^{T} q(\phi_t \mid \phi_{t-1})$, where each step $q(\phi_t \mid \phi_{t-1})$ adds Gaussian noise to ϕ_{t-1} and $q(\phi_0 \mid s)$ denotes the observed distribution of successful action parameters ϕ. The reverse process is the generative model parameterized by $\boldsymbol{\psi}$, and is similarly defined as a Markov chain: $p_{\boldsymbol{\psi}}(\phi_{T:0} \mid s) = p(\phi_T) \prod_{t=T}^{1} p_{\boldsymbol{\psi}}(\phi_{t-1} \mid \phi_t, s)$, where $p(\phi_T)$ is a standard Gaussian prior. Each step $p_{\boldsymbol{\psi}}(\phi_{t-1} \mid \phi_t, s) = \mathcal{N}(\phi_t - \epsilon_{\boldsymbol{\psi}}(\phi_t, s, t), \sigma^2 \boldsymbol{I})$ estimates the mean of a Gaussian by subtracting predicted noise given a noisy version of the action parameters, the state, and the time step. Once trained, the planner can sample from the model by simulating the reverse process $p_{\boldsymbol{\psi}}(\phi_{T:0} \mid s)$.

Separating the learning process into various diffusion models, each in charge of representing a specific distribution, as described in Section 3, reduces the difficulty of lifelong training as compared to training a single model to fit all distributions. Even so, each such diffusion model still requires continual training: a specialized sampler $p_\ell(\phi \mid \boldsymbol{x}_o)$ receives additional data as the robot uses it to attempt to solve new problems, and a generic sampler $p(\phi \mid \boldsymbol{x}_o)$ additionally receives data from new object types as the robot encounters them. Given a previous pool of data $\mathcal{Z}_{\text{old}}$ and a newly acquired set of data points $\mathcal{Z}_{\text{new}}$, we consider the following training schemes:

- **Finetuning** Starting from the previous model, train on $\mathcal{Z}_{\text{new}}$, likely forgetting previous knowledge.
- **Retraining** Train a new model on $\mathcal{Z}_{\text{new}}$ and $\mathcal{Z}_{\text{old}}$ jointly. In practice, this method has been shown to outperform all true continual training approaches, at the cost of high computational expense.
- **Replay** Starting from the previous model, train over a *balanced* sampling of data from $\mathcal{Z}_{\text{new}}$ and $\mathcal{Z}_{\text{old}}$ [7]. While more advanced strategies are possible, we found that this simple method matched the retraining performance with only 10% of the training epochs used for retraining.

5 Experimental Evaluation

Our initial experiments sought to validate that diffusion models can learn useful TAMP samplers and that the mixture distribution proposed in Section 3 achieves good performance in an offline setting. The results of these experiments set up our main evaluations over lifelong sequences of 2D and BEHAVIOR domains. Additional details are provided in Appendix A.

5.1 Visualizing Sampling Distributions

To illustrate the usefulness of our approach to learning samplers for TAMP, we created two short-horizon, 2D domains that permit us to visualize the learned distributions. The first domain (Figure 2, top) requires a robot to push a block upward. We focus on the navigation action, parameterized by the 2D coordinates, which succeeds if the block becomes reachable. However, the block can only be pushed from the bottom, so most valid actions are not conducive to success. The second domain (Figure 3, top) requires placing two blocks in an L-shaped container, where the first block fits in any of the two sections, but the second block only fits in the long section. The blocks are placed individually by selecting their 2D pose, but not all valid placements of the first block enable task success.

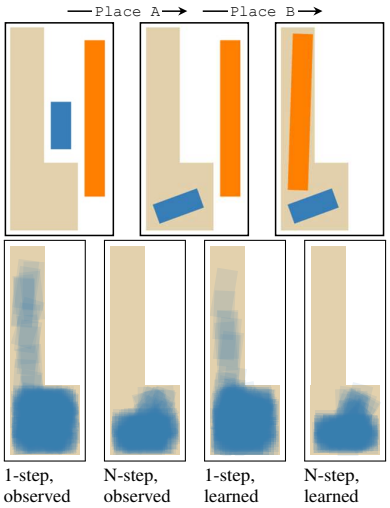


Figure 3: Top: a planner must fit two blocks in a container that restricts the longer block's placement. Bottom: distribution of parameters for achieving placement of the small block (1-step) or the overall problem (N-step), observed from a manually defined distribution and learned by diffusion models.

We collected two data sets for each domain: one with all valid actions and one with only actions that lead to task success. We then trained a diffusion-based sampler on each data set. Figures 2 and 3 (bottom) show the observed and learned distributions, demonstrating that optimizing for task success yields distributions that consider the long-term effects of actions, even if future constraints are not observable by the sampler.

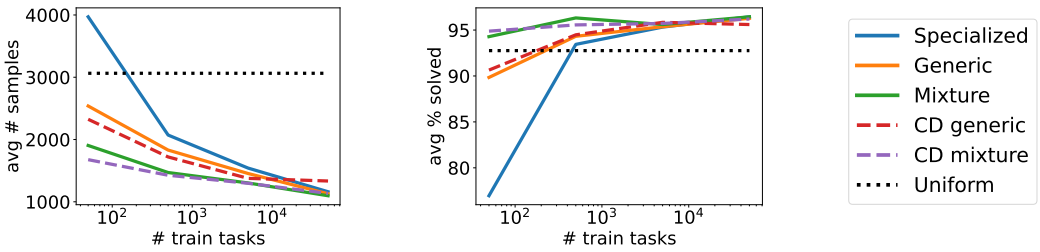
5.2 Learning Samplers from Fixed Data on 2D Domains

We next studied the impact of using our nested models in solving a variety of TAMP problems in an offline setting. We created five 2D robotics domains, each of which requires navigating, picking, and placing different objects in a target container. The tasks vary in the object size ranges, graspable regions, and container size ranges and shapes (see Appendix A.2.1 for additional details). We collected demonstration data for solving N tasks from each domain, trained the samplers on the fixed data sets, and evaluated their efficiency in solving $M = 50$ unseen tasks. We repeated the evaluation over 10 trials with various random seeds controlling the test task generation.

Our experiments considered the following diffusion-based sampler learning methods:

- Specialized: separate for each typed abstract action
- Per-domain generic: shared across abstract actions with a common controller within each domain
- Cross-domain generic: shared across abstract actions with a common controller across domains
- Per-domain mixture: our approach mixing specialized and (per-domain) generic samplers
- Cross-domain mixture: our approach mixing specialized and (cross-domain) generic samplers

Figure 5 summarizes the results for varying numbers of training tasks N. As expected, all methods become more efficient at generating good samples as they are trained on increasing amounts of data. In particular, the specialized samplers (which observe the least amount of data of all methods) are the most inefficient when trained on $N = 50$ tasks, but become as efficient as generic samplers upon training on $N = 50{,}000$ tasks. Overall, our mixtures of nested models are substantially more efficient than alternative methods, especially when trained with little data. Appendix B analyzes various mechanisms for selecting between specialized and generic samplers, demonstrating that using geometric auxiliary signals as a proxy for sampler generalization is a strong choice.



(a) Number of samples per solved problem (b) Number of problems solved

Figure 5: Results of learning diffusion models as TAMP samplers from offline data. Given sufficient data, all samplers solve the majority of problems efficiently. In the small-data regime (note the log-scale of the x-axis), sharing data across samplers improves sampling efficiency. The mixture distributions learned either individually on each domain (Mixture) or across all domains (CD mixture) are best, thanks to their ability to automatically select generic or specialized samplers during planning.

5.3 Lifelong Evaluation on 2D Domains

We now proceed to describe our main results in the lifelong learning setting. Our evaluation presented the agent with a sequence of problems from each domain in turn (first domain 1, then domain 2...), imposing a nonstationary distribution. The agent attempted to solve each problem given its current sampler, and subsequently used the collected data for additional training. There was no separate test phase: the agent's performance was assessed over its attempts to solve the problems in the sequence (before training on them). We updated the models of each method using the retraining, finetuning, and replay strategies described in Section 4.

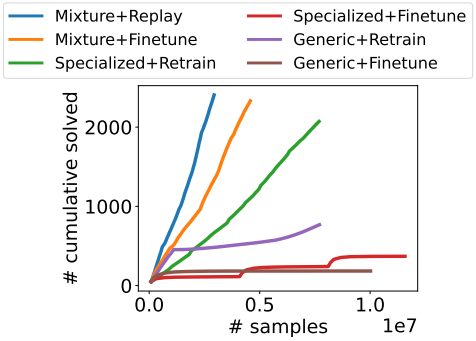


Figure 4: Lifelong learning results on 2D domains. The mixture distributions are vastly superior. Finetuning the models directly fails due to forgetting, especially with generic and specialized samplers.

Figure 4 shows the cumulative number of problems solved against the number of generated samples for each method. Our mixture sampler is substantially more efficient at solving problems than either of the baseline methods. In the lifelong setting, agents use their current version of the samplers to collect data, which explains why even the retraining version of the generic baseline fails so catastrophically. The agent uses the model trained on previous types to generate samples for novel object types; early on, those samplers are overfit to the small set of initial types, causing them to fail to solve any problems and therefore preventing the robot from acquiring any useful data for subsequent updates. Specialized samplers instead work reasonably well, but are slowed down by their inability to quickly generate samples when trained over little data. To demonstrate the importance of avoiding forgetting in this context, we evaluated finetuning variants of each approach, which is known to suffer from forgetting. Clearly, the forgetful approach performs noticeably worse than the proper lifelong mechanisms, demonstrating the need to retain knowledge throughout the task sequence.

5.4 Lifelong Evaluation on BEHAVIOR

We next applied the evaluation protocol of Section 5.3 to 10 families of BEHAVIOR tasks (see Appendix A.3 for details). Figure 6 demonstrates that our method enables continual learning over this more complex and realistic domain. Notably, the agent improves even over the hand-crafted samplers that we provided as a starting point, demonstrating the usefulness of our approach even when engineered solutions are already in place.

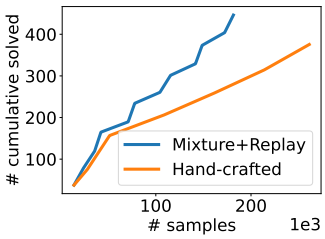


Figure 6: Lifelong learning results on BEHAVIOR. Our lifelong learner progressively improves over its initial (hand-crafted) samplers, becoming increasingly better at solving diverse problems.

6 Related Work

Lifelong learning The literature on lifelong or continual learning has grown rapidly in recent years, primarily focused on avoiding catastrophic forgetting [8] in supervised learning settings [9, 10, 11, 12, 13]. In particular, a number of techniques replay past data to ensure knowledge retention [14, 15, 7], which we adopt. Our focus is on enabling a robot system to become increasingly capable over time via compositionality. Few works have studied compositionality in lifelong supervised [16, 17, 18] and reinforced [19, 20, 21] domains. Prior lifelong learning work considers training an agent over a sequence of machine learning tasks and subsequently (in a separate phase) evaluating the system over those same tasks. We propose a more natural evaluation setting without train/test splits, where the agent continually seeks to solve TAMP problems and uses the data from those problems to learn. While seemingly this formulation does not require forgetting avoidance (since previous problems are never encountered again), prior work has suggested that knowledge retention is also useful for generalization to *future* problems [2]. Our results in Sections 5.3 and 5.4 exemplify this notion.

Learning for TAMP Numerous recent methods seek to broaden the capabilities of TAMP systems beyond engineered solutions to specific robot problems by leveraging learning techniques. A majority of such methods focus on symbolic aspects of the problem: given a partial symbolic description of a domain, use demonstration data to fill the gaps in symbolic space to enable solving additional problems. This can take the form of learning operators (i.e., action abstractions) [22, 23, 5] or predicates (i.e., state abstractions) [24, 25, 26]. Given the difficulty of mapping abstract actions to continuous robot actions, our focus is on learning at the continuous level, specifically in the form of samplers. The most closely related approach, which learns samplers for SeSamE planners, considers a very simple class of regression samplers with a learned rejection classifier [5]. Other methods for learning TAMP samplers use more powerful generative models, but not diffusion models [27, 28, 29, 30, 31]. None of these prior works study the underlying multitask problem, nor do they operate in a lifelong setting as our method does. Other (less) related work uses reinforcement learning to bridge between the discrete and continuous actions [32, 33] or automatically decomposes motion plans into discrete components using vector quantized variational autoencoders [34].

7 Limitations

Our experiments apply our lifelong sampler learning approach to a SeSamE strategy, and so we require access to a faithful simulator of the environment. While we can often simulate a robot's own state accurately, simulating environments with many objects is difficult. In such cases, we would prefer to use a different TAMP method; we leave this line of work open for future investigation. One potential direction to improve our learned samplers is to develop more deliberate exploration strategies. Our method directly uses the learned models to generate samples; an approach that trades off exploration and exploitation, considering the learning potential of each sample, could yield additional benefits. Moreover, our generative models are conditioned only on the objects involved in a given action. Distributions conditioned on all other objects or on the future actions the robot may execute could generate samples that are even better geared toward overall success. Additionally, the lifelong training method we use requires replaying all past data. In our experiments, this represents only a minuscule portion of the computation (which is overwhelmed by planning). In longer deployments where the data may grow arbitrarily, it would not be feasible to replay all past data, so subsampling strategies—which our approach can trivially adopt—should be employed.

8 Conclusion

We proposed a novel formulation of the embodied lifelong learning problem for TAMP systems, which emphasizes realistic evaluation of the robot as it attempts to solve tasks over its lifetime. Our solution approach learns a mixture of nested generative models, assigning higher weight to models that attain low error on auxiliary prediction tasks. Our experiments on 2D and BEHAVIOR domains demonstrate the ability of our approach to acquire knowledge over a lifetime of planning.

References

[1] S. Srivastava, C. Li, M. Lingelbach, R. Martín-Martín, F. Xia, K. E. Vainio, Z. Lian, C. Gokmen, S. Buch, K. Liu, S. Savarese, H. Gweon, J. Wu, and L. Fei-Fei. BEHAVIOR: Benchmark for everyday household activities in virtual, interactive, and ecological environments. In *Proceedings of the 5th Conference on Robot Learning (CoRL-22)*, pages 477–490, 2022.

[2] J. A. Mendez and E. Eaton. How to reuse and compose knowledge for a lifetime of tasks: A survey on continual learning and functional composition. *Transactions on Machine Learning Research (TMLR)*, 2023.

[3] C. R. Garrett, R. Chitnis, R. Holladay, B. Kim, T. Silver, L. P. Kaelbling, and T. Lozano-Pérez. Integrated task and motion planning. *Annual Review of Control, Robotics, and Autonomous Systems*, 4(1):265–293, 2021.

[4] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL: The planning domain definition language. *Technical report, Yale Center for Computational Vision and Control.*, 1998.

[5] R. Chitnis, T. Silver, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Learning neuro-symbolic relational transition models for bilevel planning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-22)*, pages 4166–4173.

[6] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems 33 (NeurIPS-20)*, pages 6840–6851, 2020.

[7] 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.

[8] M. McCloskey and N. J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of Learning and Motivation*, volume 24, pages 109–165. Elsevier, 1989.

[9] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences, PNAS*, 114(13):3521–3526, 2017.

[10] F. Zenke, B. Poole, and S. Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning, ICML-17*, pages 3987–3995, 2017.

[11] Z. Li and D. Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence, TPAMI*, 40(12):2935–2947, 2017.

[12] C. V. Nguyen, Y. Li, T. D. Bui, and R. E. Turner. Variational continual learning. In *6th International Conference on Learning Representations, ICLR-18*, 2018.

[13] J. Serrà, D. Surís, M. Miron, and A. Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *Proceedings of the 35th International Conference on Machine Learning, ICML-18*, pages 4548–4557, 2018.

[14] D. Lopez-Paz and M. Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems 30, NIPS-17*, pages 6467–6476, 2017.

[15] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. Efficient lifelong learning with A-GEM. In *7th International Conference on Learning Representations, ICLR-19*, 2019.

[16] J. A. Mendez and E. Eaton. Lifelong learning of compositional structures. In *9th International Conference on Learning Representations, ICLR-21*, 2021.

[17] T. Veniat, L. Denoyer, and M. Ranzato. Efficient continual learning with modular networks and task-driven priors. In *9th International Conference on Learning Representations, ICLR-21*, 2021.

[18] O. Ostapenko, P. Rodriguez, M. Caccia, and L. Charlin. Continual learning via local module composition. In *Advances in Neural Information Processing Systems 34, NeurIPS-21*, pages 30298–30312, 2021.

[19] E. Brunskill and L. Li. PAC-inspired option discovery in lifelong reinforcement learning. In *Proceedings of the 31st International Conference on Machine Learning, ICML-14*, pages 316–324, 2014.

[20] C. Tessler, S. Givony, T. Zahavy, D. Mankowitz, and S. Mannor. A deep hierarchical approach to lifelong learning in Minecraft. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, AAAI-17*, 2017.

[21] J. A. Mendez, H. van Seijen, and E. Eaton. Modular lifelong reinforcement learning via neural composition. In *10th International Conference on Learning Representations, ICLR-22*, 2022.

[22] T. Silver, R. Chitnis, J. Tenenbaum, L. P. Kaelbling, and T. Lozano-Pérez. Learning symbolic operators for task and motion planning. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-21)*, pages 3182–3189, 2021.

[23] N. Kumar, W. McClinton, R. Chitnis, T. Silver, T. Lozano-Pérez, and L. P. Kaelbling. Overcoming the pitfalls of prediction error in operator learning for bilevel planning. *arXiv preprint arXiv:2208.07737*, 2022.

[24] J. Loula, K. Allen, T. Silver, and J. Tenenbaum. Learning constraint-based planning models from demonstrations. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-20)*, pages 5410–5416, 2020. doi:10.1109/IROS45743.2020.9341535.

[25] A. Curtis, T. Silver, J. B. Tenenbaum, T. Lozano-Pérez, and L. Kaelbling. Discovering state and action abstractions for generalized task and motion planning. In *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI-22)*, volume 36, pages 5377–5384, 2022.

[26] T. Silver, R. Chitnis, N. Kumar, W. McClinton, T. Lozano-Pérez, L. P. Kaelbling, and J. Tenenbaum. Predicate invention for bilevel planning. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)*, 2023.

[27] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez. Learning compositional models of robot skills for task and motion planning. *International Journal of Robotics Research*, 40 (6–7):866–894, jun 2021.

[28] Z. Wang, C. R. Garrett, L. P. Kaelbling, and T. Lozano-Pérez. Active model learning and diverse action sampling for task and motion planning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-18)*, pages 4107–4114, 2018.

[29] J. Ortiz-Haro, J.-S. Ha, D. Driess, and M. Toussaint. Structured deep generative models for sampling on constraint manifolds in sequential manipulation. In *5th Annual Conference on Robot Learning (ICLR-21)*, 2021.

[30] B. Kim, L. Kaelbling, and T. Lozano-Pérez. Guiding search in continuous state-action spaces by learning an action sampler from off-target search experience. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[31] B. Kim, L. Shimanuki, L. P. Kaelbling, and T. Lozano-Pérez. Representation, learning, and planning algorithms for geometric task and motion planning. *The International Journal of Robotics Research*, 41(2):210–231, 2022.

[32] T. Silver, A. Athalye, J. B. Tenenbaum, T. Lozano-Pérez, and L. P. Kaelbling. Learning neuro-symbolic skills for bilevel planning. In *6th Annual Conference on Robot Learning (CoRL-22)*, 2022.

[33] G. Liu, J. de Winter, D. Steckelmacher, R. K. Hota, A. Nowe, and B. Vanderborght. Synergistic task and motion planning with reinforcement learning-based non-prehensile actions. *IEEE Robotics and Automation Letters (RA-L)*, 8(5):2764–2771, 2023.

[34] J. J. Johnson, A. H. Qureshi, and M. Yip. Learning sampling dictionaries for efficient and generalizable robot motion planning with transformers. *arXiv preprint arXiv:2306.00851*, 2023.