Lifelong Robot Library Learning: Bootstrapping Composable and Generalizable Skills for Embodied Control with Language Models Paper Review

Paper Review by Tyler Kim

Summary

The paper *Lifelong Robot Library Learning: Bootstrapping Composable and Generalizable Skills for Embodied Control with Language Models* by Georgio Tziafas and Hamidreza Kasaei introduces a new approach to use large language models (LLM) to generate robot policies. Current advances in robot learning for learning multimodal policies required a plethora of data and used gradient descent for optimization causing difficulty in scaling and hinders applicability in lifelong learning. In addition, previous LLM applications used hand-crafted prompt engineering limiting the agent to a certain range of tasks. Therefore, the paper proposes an LLM-based lifelong learning agent named Lifelong Robot Library Learning (LRLL) which generates policy code, explores tasks in a simulation, and expands the skill repertoire of the robot.

The LRLL model uses a wake-sleep optimization technique where during the wake phase, a human provides a demonstration and a hint of which the agent essentially simulates and verifies the proposed tasks in a simulator and stores it into memory. During the sleep phase, the agent reflects upon the experience by clustering it with similar experiences, abstracting it to create new skills, and stores it into the skill library. The skill library holds the skills that the agent has learned, more specifically, skills are stored as a python API function in the library. The output of the agent is python code. In addition a pre-trained vision model was implemented for zero-shot vision-language grounding using the CLIP and MDETR models.

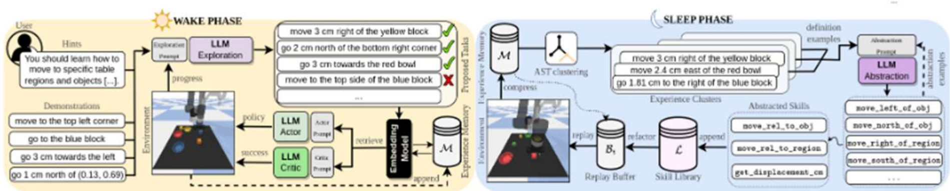

Fig. 2: Overview of an *LRLL* learning cycle. At the beginning of the wake phase, a human user provides demonstrations and hints, out of which an LLM-based *exploration* module proposes tasks to complete, while an LLM-based *actor-critic* agent interacts with the environment to execute and verify tasks. During sleep, the experiences are clustered according to their code's abstract syntax trees (AST) and distilled into new skills with an LLM *abstractor*. The new skills refactor the acquired experiences, which are replayed in the environment in order to compress the memory. We note that for brevity purposes, we omit showing the actor-critic modules during replay at the sleep phase. We illustrate examples from our first curriculum cycle (*spatial coordination*).

The paper used the blocks-and-bowls setup from previous works and organized them into four categories: spatial coordination, visual reasoning, object manipulation, and rearrangement. To evaluate the model the experiment was divided into three categories: seen instructions with seen attributes (SA), seen instructions with unseen attributes (UA), and unseen instructions with unseen attributes (UI). The experiment also proposed two more splits to test whether the agent can indeed transfer knowledge between tasks (forward-transfer (FT)) and to what extent the agent forgets a task (backward-transfer (BT)). The baseline models used were CLIPort, LLM's given static prompts (LLM-static), LLRL without sleep (LLRL-no-sleep), and LLRL without wake (LLRL-no-wake).

The first test was on a simulated tabletop scenario where CLIPort struggled with unseen attributes, LLM-static is robust to unseen attributes but generates non-executable code in later cycles, and LLRL had an average increase of about 6% for unseen attributes and about 10% for unseen instructions compared to LLM-static.

TABLE I: Averaged success rates (%) over seen instructions with seen/unseen attributes (SA/UA) and unseen instructions (UI), organized in a 4-cycle curriculum with 10 trials per instruction. Best results are in bold.

Tasks	CLIPort [27]			LLM-static [15]			LRLL (ours)		
	SA	UA	UI	SA	UA	UI	SA	UA	UI
Spatial Coord/on	-	-	-	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**
Visual Reasoning	-	-	-	90.0	83.3	66.6	**91.7**	**94.0**	**85.1**
Object Manip/on	**98.3**	37.1	4.1	95.0	94.1	80.0	98.1	**98.9**	**90.4**
Rearrangement	70.8	13.9	0.3	93.0	90.0	60.6	**97.0**	**95.4**	**70.9**
Average	84.5	25.5	2.2	94.5	91.9	76.8	**96.9**	**97.1**	**86.6**

The second study was an ablation study where the average success rate for all baselines were compared in the FT/BT instructions. No baseline reported an increase in the BT test but were robust in tasks and only LLRL and LLM-static had an increase in averaged success for FT. Prompt saturation rendered the LLM-static unstable, LLRL-no-wake performed lower across cycles, LLRL-no-sleep requires retrieval of many examples to get sufficient context, and LLRL could maintain high performance in UI tasks with minimal retrievals.

TABLE II: Averaged success rates (%) over unseen task combinations (FT) and a subset of previous task combinations (BT) for each cycle and baseline. Best results are in bold.

Tasks	LLM-static [15]		LRLL-no-wake		LRLL-no-sleep		LRLL	
	FT	BT	FT	BT	FT	BT	FT	BT
Spatial Coord/on	**100.0**	**100.0**	68.7	60.0	**100.0**	**100.0**	**100.0**	**100.0**
Visual Reasoning	60.0	**100.0**	40.0	55.0	**80.0**	**100.0**	**80.0**	**100.0**
Object Manip/on	55.3	60.0	46.6	40.4	78.4	80.0	**94.0**	**80.0**
Rearrangement	45.7	55.3	50.0	46.6	64.0	78.4	**70.2**	**94.0**
Average	65.1	78.9	51.3	50.5	80.6	89.6	**88.1**	**93.5**

The paper repeated the curriculum with LRLL for a dual-arm robot and reported that the robot did well and most errors derived from collisions and perception errors.

The major contributions of this paper are the LRLL framework that has dynamic storage and retrieval of past experiences for context, self-guided exploration policy, skill abstractor that distills recent experiences into new library skills, and allows minimal human intervention to bootstrap new skills with minimal interaction.

Strengths

Lifelong Robot Library Learning: Bootstrapping Composable and Generalizable Skills for Embodied Control with Language Models provide a couple of strengths. One strength is that the LRLL model generalizes better than previous approaches, in particular, it is able to develop new skills from already learnt skills. The LRLL is memory efficient refactoring code when storing it into memory. The LRLL is scalable, able to learn new skills. There are a couple of things I learned from this paper. I learned about a new approach for optimization using the wake-sleep cycle approach, a way to cluster programs together using the abstract syntax tree, and a little bit about multi-agent simulations.

Potential Improvements

This paper also had a couple of weaknesses that hindered it from getting closer to its stated goal or contribution. One weakness that the paper had was that the provided website did not have any information at all about the paper and links were often missing. Another weakness of this paper was that it did not provide any examples of how LRLL actually works. The provided figures seem to provide a partial understanding of the LRLL in the context of an actual scenario. Other weaknesses regarding the LRLL framework include a limited choice vision API, skills can only be expressed symbolically, initial prompts required exploration/abstraction modules changing domains or LLM engines, and it was costly. A weakness that hindered the paper from leveraging the latest techniques or models to improve the method was the use of the vision API. Although at the time of their experiment, GPT-4o was not yet released, I think GPT-4o could have made a significant difference in the project. Another weakness could be that the LRLL uses less efficient APIs.

Extensions

Extensions of this paper could include investigating other vision-language APIs for their agent including GPT-4o and others. Another extension for the paper could be to use the LRLL for a more complicated robot just a humanoid hand where the primitive functions could be the individual joints of the humanoid hand. Another idea would be to try to use visual instructions and demonstrations to teach the agent new skills.