Synthetic Manipulation Data Generation using Large Language Models to Train Real-World Language-Conditioned Robotic Manipulation Agents

Akshay Muppidi
Guilderland High School
Guilderland, NY, USA

***Abstract*—The advancement of robotic manipulation in dynamic environments has been significantly bolstered by the integration of language-conditioned goal-based manipulation AI agents, notably through agents such as CLIPort. This paper introduces a novel approach to overcoming one of the field's major challenges: the scarcity of diverse, high-quality training data. By leveraging the capabilities of the open-source large language model (LLM), CodeLlama-Instruct, I create a pipeline to autonomously generate a vast array of task ideas and corresponding demonstrations, thereby expanding the existing dataset by 15x. This methodology not only democratizes the development process by reducing reliance on manual dataset creation but also significantly enhances the model's performance on both familiar and novel tasks. Through a series of experiments, I demonstrate that models pre-trained on these LLM-generated augmented datasets exhibit substantial improvements in performance, with up to 52.99% increase in success rates in simulation tasks and up to 146.10% in real-world tasks using the XArm-7 robot equipped with a suction gripper. These results not only underscore the effectiveness of integrating LLM-generated tasks into training datasets but also highlight the potential for these models to adapt and perform in a wide range of real-world applications. This study marks a pivotal step towards more autonomous, versatile, and efficient robotic systems, paving the way for future advancements in robotic manipulation.**

***Index Terms*—Robotic Manipulation, Imitation Learning, Large Language Models, Suboptimal Demonstrations.**

INTRODUCTION

The field of robotic manipulation has been rapidly evolving, particularly in the realm of language-conditioned goal-based manipulation, which integrates linguistic instructions with physical actions in dynamic environments [2, 3]. A pioneering development in this area is CLIPort in the work "CLIPort: What and Where Pathways for Robotic Manipulation" [5]. This system synergizes the broad semantic understanding of Contrastive Language-Image Pre-Training (CLIP) [4] with the spatial precision of Transporter Networks, a concept presented in "Transporter Networks: Rearranging the Visual World for Robotic Manipulation" [7]. Both these approaches represent significant strides in enabling robots to interpret and manipulate objects based on abstract concepts and fine-grained spatial awareness.

The core challenge addressed by CLIPort is language-conditioned goal-based manipulation, the manipulation of objects based on visual observations and language instructions. For example, a task might involve arranging disks on a tabletop as per a specific instruction, like "stack the red disk on the yellow, the orange on the red, and the blue on the orange." The learning process of CLIPort leverages imitation learning [1, 6], akin to the methodologies in Transporter Networks, where the agent is trained on expert demonstrations aligned with goal-oriented texts. This approach, however, reveals an inherent limitation: the availability of diverse, high-quality training data. Expert-generated demonstrations are labor-intensive and scarce, leading to a bottleneck in data quantity and variety, which are crucial for training robust and versatile agents.

This project addresses this limitation by using LLMs for program synthesis, creating new task ideas and demonstrations for language-conditioned goal-based manipulation. The potential of LLMs to generate synthetic yet realistic demonstrations opens the door to a more extensive dataset, enabling the exploration of new research questions. Such an expansion in training data is critical, not only for enhancing the agent's performance but also for examining the influence of data quality and diversity in its learning process.

This project creates a custom pipeline that employs an open-source LLM, CodeLLama-Instruct, to create new task ideas and generate code required to demonstrate that task. Using this pipeline, I augment the original dataset with LLM generated demonstrations. I show that by training on the LLM augmented dataset, I see significant improvement in the evaluation of both seen and unseen (zero-shot) tasks in both simulation as well as real-world robotic arm manipulation.

Trade-offs

Traditional methods, limited by datasets derived solely from expert demonstrations, fail to capture the diversity of real-world scenarios, thus restricting a learning agent's generalizability.

The strategic use of LLMs to generate a wide array of task scripts addresses this limitation effectively. LLMs like CodeLlama-Instruct facilitate the creation of extensive and varied task scripts, allowing dataset augmentation beyond manual capabilities. This is not only beneficial in terms of quantity but also crucial for introducing a diverse array of scenarios to the learning agent.

It's important to acknowledge that CodeLlama may not match other expensive LLMs such as GPT-4 in coding proficiency and might require multiple iterations for successful script generation. But Opting for an open-source LLM like CodeLlama-Instruct 13B for script generation reflects my commitment to accessibility in robotics research. Although, expensive GPT-4-Turbo models are more robust and effective, my focus on open-source and cost-effective alternatives broader the mission of promoting open foundation models. Additionally, compared to the rapid output of OpenAI's API, CodeLlama's script generation can be slower, necessitating a GPU with sufficient memory to handle 32k context inputs and the 13B parameters. Despite these limitations, the free usage of CodeLlama offers a significant advantage. It allows for extensive experimentation with prompts without the financial burden often associated with high-cost models. This freedom to iterate and refine without cost concerns is a key factor in my decision to utilize CodeLlama-Instruct.

Key Results

A standout feature of my approach is the significant reduction of human labor traditionally required for generating task datasets in robotic manipulation. By leveraging CodeLlama-Instruct, I autonomously generated a diverse array of new task ideas and corresponding scripts, effectively removing the bottleneck associated with manual dataset creation. This innovation **led to a 15-fold increase** in the size of the dataset, significantly enhancing the variety and complexity of tasks available for training.

The autonomous generation of task datasets not only streamlines the process of dataset expansion but also ensures a higher degree of diversity and complexity in the tasks, often challenging to achieve through human-generated content alone. This methodology democratizes the development of advanced robotic manipulation systems by significantly lowering the barriers to entry related to data collection and curation. It allows researchers and developers to focus more on model development and refinement rather than on the labor-intensive task of dataset creation.

Moreover, the significant improvements observed in CLIPort's performance through this autonomous dataset generation process are noteworthy. Specifically, **I observed performance increases of up to 52.99% in simulation tasks and up to 146.10% in real-world tasks with the XArm-7 robot equipped with a suction gripper, when comparing models trained on augmented datasets against the baseline.** These improvements underscore the quality and applicability of the autonomously generated datasets, which not only matched but, in many instances, surpassed the effectiveness of manually curated datasets.

Methods and Approach

LLM Prompting and Task Generation

My approach is leveraging CodeLlama-Instruct 13B for task idea generation and script synthesis involved constructing comprehensive prompts. These prompts aimed to exploit the model's expansive knowledge and its capacity for innovation.

Task Idea Generation Prompt Structure: The initial stage involved formulating prompts to inspire the generation of new, feasible tasks for robotic manipulation. These prompts were designed to elicit creative and diverse task scenarios that extend beyond the current limitations of existing datasets.

- **Introduction to Robotics Context:** The prompt began by contextualizing the request, outlining my objective to enrich a dataset with novel tasks for a robot trained on language commands.
- **Desired Task Characteristics:** I specified the desired attributes of tasks, including their complexity and environmental setup, guiding the LLM towards producing relevant and challenging tasks.
- **Incorporation of Ravens Examples:** Examples of current tasks from the Ravens benchmark dataset were included to serve as a creative springboard, demonstrating the expected level of detail and innovation for the new tasks.
- **Request for Task Ideas:** The prompt concluded with a direct appeal for novel task ideas, emphasizing the need for inventive combinations of actions and objects that adhere to complex language instructions.

Task Code Synthesis Prompt Structure: Transitioning from ideation to the synthesis of executable task scripts required adjusting the prompts to focus on translating abstract task ideas into Python scripts suitable for my simulation environment.

- **Task Idea Recap:** Each code synthesis prompt opened with a summary of the proposed task idea, ensuring its core concept was emphasized.
- **CLIPort Environment Overview:** A brief overview of the CLIPort API and simulation environment set the stage, detailing the available actions and objects to ensure compatibility.
- **Task Requirements:** Detailed requirements for the Python script were outlined, specifying object setup, action sequences, and success conditions.
- **Python Script Request:** The main request asked CodeLlama to generate the task script, encouraging the use of comments for clarity and script organization.
- **Execution and Error Capture:** Upon generation, scripts were executed within the PyBullet simulation environment to validate their functionality. This crucial step ensured that each script not only met syntactical standards but was also viable for robotic task execution.
- **Iterative Error Feedback Integration:** In instances where scripts encountered execution failures or syntax

errors, these errors were meticulously cataloged. The specific error messages generated by PyBullet were then fed back into the prompt for CodeLlama-Instruct 13B. This feedback explicitly highlighted the issues encountered in previous iterations, urging the model to adjust its output to avoid similar pitfalls. Common errors, identified through their frequent occurrence across multiple script generations, were cached. This repository of common errors became a permanent feature of every task generation prompt, serving as a preemptive guide to avoid known issues. Should a script fail to reach the required standard of functionality after the allotted 20 attempts, my pipeline initiates a fallback protocol. This involves returning to the task idea generation phase, where CodeLlama-Instruct 13B is prompted to conceive a new task idea, followed by a fresh attempt at script generation.

A visual overview of the pipeline is seen in Figure 1. Examples of newly created task ideas and their scripts include 'sort-colored-balls', 'arrange-bowls-in-line', 'align-ball-in-colored-bowl' (**Fig 2 a**), and 'build-tree' (**Fig 2 b**, animation).

Demonstrations and Environment

Post-script generation, a manual review ensured scripts accurately fulfilled task descriptions. Most scripts required minimal adjustments after passing runtime and execution checks.

My simulation utilized PyBullet, paired with a UR5 robot model, aligning with the Ravens setup for tabletop manipulation. This environment, integrated with CLIPort, facilitated the use of Ravens' scripts as one-shot learning data.

An example script for the `sorted-colored-balls` task generated by my pipeline is available for reference here.

Experimental Setup

In the Experimental Setup section of my study, I meticulously designed a series of experiments to explore three pivotal questions concerning the integration of LLM-generated tasks into the training regimen of robotic manipulation policies. Firstly, I examine whether the incorporation of tasks generated by my pipeline enhances the generalization capabilities of these policies. This investigation is crucial in understanding the breadth of learning and adaptability that can be achieved through synthetic task training. Secondly, I delve into the potential incremental benefits of increasing the volume of LLM-generated tasks included in the training process. The hypothesis here is that a larger dataset might provide a richer learning experience, thereby contributing to a more robust policy performance across a wider array of tasks. Thirdly, and perhaps most significantly from a practical standpoint, I assess the translatability of the gains from LLM-augmented training in simulation environments to real-world robot policy deployment. This part of the study is designed to evaluate the feasibility and effectiveness of simulation-based training enhancements in actual robotic operations, bridging the gap between simulated training environments and real-world applications.

Demonstrations

For the newly created LLM-generated tasks and the existing Ravens benchmark tasks, I produced 150 expert demonstrations each for each of the 150 new and 10 human-engineered tasks.

Evaluation Protocol

CLIPort's performance was evaluated on the original 10 Ravens benchmark tasks. See Table I for an overview of the 10 benchmark tasks, and whether the task requires precise placement of objects to their target pose for completion, and whether the task is multi-step, meaning there is a specific sequence of manipulation steps that need to be taken by the agent for successful completion. Each task's success was measured based on a predefined metric (from the Ravens dataset) and averaged over 15 trials.

A critical aspect of my evaluation protocol involves assessing CLIPort's performance on tasks involving unseen colors, shapes, and objects. This approach is designed to test the model's generalization capabilities. During training, the model is exposed to a specific set of colors, shapes, and object types. However, during evaluation, it is required to manipulate objects that feature a completely new set of attributes (colors, shapes, or object types) that it has not encountered during training. Note the only task I didn't apply this methodology on is for align-rope as there was no "unseen" version of the task provided in the dataset. For detailed descriptions, success metrics, and "unseen" evaluations for benchmark tasks I refer the reader to the original CLIPort and transporters papers (although I outline an example of a benchmark task and its success metric below).

TABLE I
Tasks and Attributes

Task	Precise Placement	Multi-step
assembling-kits-seq-unseen-colors	✓	✓
packing-box-pairs-unseen-colors	✓	✓
packing-unseen-google-objects-group	X	✓
packing-unseen-google-objects-seq	X	✓
packing-unseen-shapes	X	X
put-blocks-in-bowl-unseen-colors	X	X
separating-piles-unseen-colors	✓	✓
stack-block-pyramid-seq-unseen-colors	✓	✓
towers-of-hanoi-seq-unseen-colors	✓	✓
align-rope (seen)	✓	✓

To help understand my evaluation approach, I provide a full description of the packing-unseen-shapes task from the benchmark dataset.

packing-unseen-shapes task (Fig 4): The objective is to place a specified shape in a brown box, surrounded by four distractor shapes. The color of these shapes is randomized but irrelevant to the task's objective. This task tests the agent's semantic understanding of arbitrary shapes without requiring precise placements. The task is trained with shapes that the model has seen during training but evaluated on entirely new,

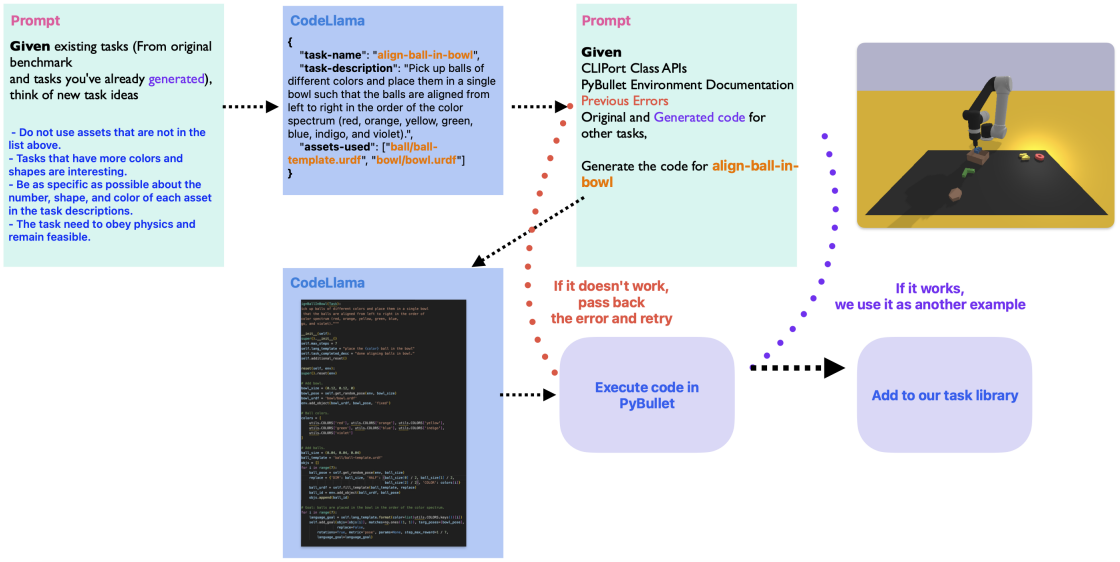

Fig. 1. A visual overview of my proposed pipeline for creating diverse LLM-generated manipulation datasets

Align-ball-in-colored-bowl and Build-tree tasks

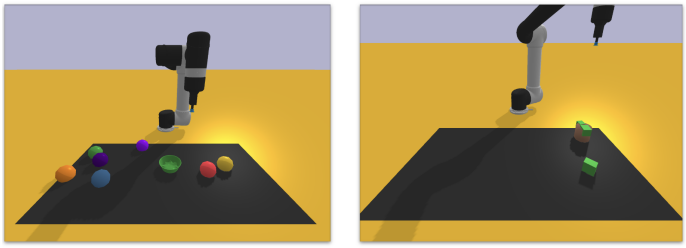

Fig. 2. **(a)** In the align-ball-in-colored-bowl task, the language goal is to "pick up the blue ball and place it in the green bowl." **(b)** In the build-tree task, the language goal is "construct a simple tree. Start by placing a brown cylinder upright to form the tree trunk. Then, arrange the green blocks on top of the cylinder to represent the leaves of the tree." **Both task ideas were generated using CodeLlama.**

Assembling-kits task.

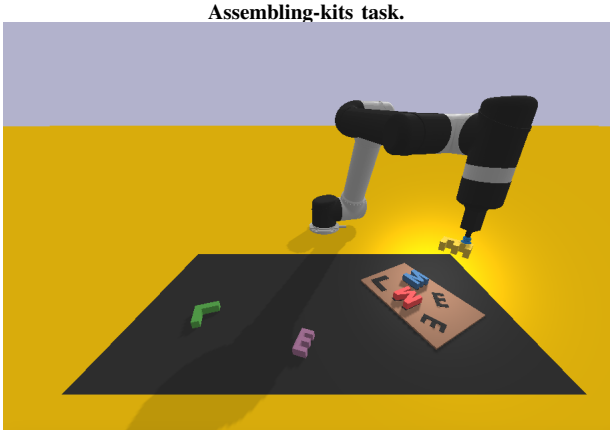

Fig. 3. The assembling-kits task requires the manipulator to precisely place each specified shape in the specified hole following the order prescribed in the language instruction at each timestep.

unseen shapes. Success is determined by whether the correctly identified shape is placed within the bounds of the brown box.

Packing-unseen-shapes task.

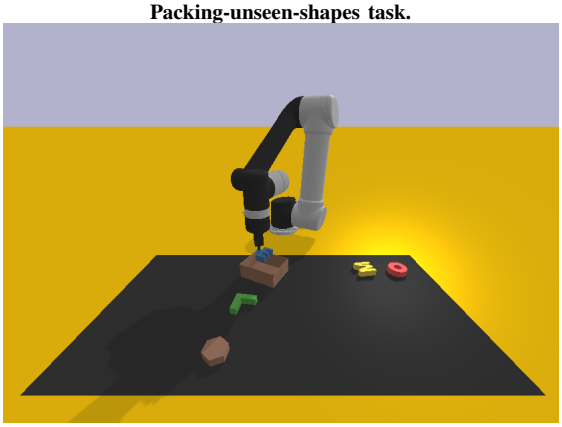

Fig. 4. In this case of the packing-unseen-shapes task, the blue E shaped object is asked to be placed in the brown box. Note that this task does not require precise placement.

Experiments Conducted

My study was methodically designed to evaluate the enhancement in policy generalization and real-world applicability after training on tasks generated by LLM. The experimental framework encompassed the following stages:

- **Baseline Model Training:** Initiated with training a baseline model exclusively on the 10 traditional tasks from the Ravens benchmark, setting a benchmark for performance comparison.
- **LLM-Generated Task Pretraining:** To understand the dataset size impact on model performance, models underwent pretraining with an incremental set of new tasks

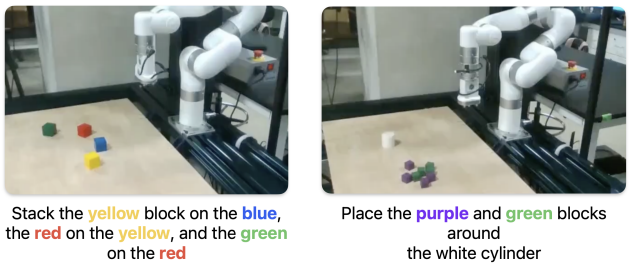

Fig. 5. Real-world language-conditioned goal-based manipulation set up at Harvard Computational Robotics Lab.

generated by CodeLlama-Instruct 13B, namely 50, 100, and 150 tasks.

- **Simulation-Based Evaluation:**
  1) Evaluated the pretrained models against the original Ravens benchmark evaluation tasks in a simulation environment, assessing generalization capabilities.
  2) Conducted zero-shot task evaluations with completely new prompts, not related to training data, to test adaptability to novel scenarios.
- **Real-World Evaluation:**
  1) Applied similar evaluation protocols as in simulation to the real-world setting, using the XArm-7 robot equipped with a suction cup, housed at the Harvard Computational Robotics Lab. 5
- **Evaluation Metric:** Adoption of the 0 to 100 success scoring system from the Ravens benchmark for performance quantification. The evaluation metric was consistently applied across all policy parameterizations.
- **Computational and Physical Resources:** The data generation, alongside the training processes, leveraged the computational power of multiple GPUs at the Harvard Computational Robotics Lab Cluster. The real-world evaluations, specifically the baseline and 100 epochs pretrained models, incorporated a limited set of real-world data due to restricted robot access, supplemented with data augmentations and model fine-tuning.

RESULTS

A. Full Ravens Simulation

In the assessment conducted on the full Ravens benchmark in simulation, the CLIPort model and its variations pretrained on LLM-generated tasks demonstrated notable performance improvements. Here, ”pretrained” refers to models that were initially trained on a combination of the original training dataset plus an augmented set of tasks generated by CodeLlama-Instruct. Specifically, models were pretrained on datasets augmented with 50, 75, and 150 LLM-generated tasks, in addition to the original training dataset. The models exhibited average success rate increases of 15.52%, 31.98%, and 52.99%, respectively, for each augmented dataset size (Figure 6) compared to the baseline. This trend indicates a direct correlation between the amount and diversity of training tasks and the model's increased ability to interpret and execute the given instructions within simulated environments.

B. Zero-Shot Generalization in Simulation

For the zero-shot generalization experiments, a different approach was employed. To accurately test the models' ability to generalize to completely new tasks, the original training dataset was modified by systematically omitting 5, 7, and eventually 10 tasks that were then used as unseen tasks for evaluation. This methodology ensures that the zero-shot generalization performance is a true reflection of the models' ability to apply learned skills to entirely new situations without prior direct exposure. The models pretrained on 50, 75, and 150 LLM-augmented tasks (excluding the omitted tasks from the original dataset for true zero-shot evaluation) showcased remarkable improvements in handling unseen tasks, with increases in success rates of 168.29%, 279.64%, and 392.50%, respectively (Figure 7) compared to baseline. These results underscore the significant advantage of pretraining on diverse datasets in enhancing the models' zero-shot generalization capabilities.

C. Real-World Evaluation with XArm-7 robot equipped with a suction gripper

In the real-world tests using the XArm-7 robot, the benefits of LLM-augmented pretraining were consistently evident. With success rate improvements over the baseline method of 87.70% for the model pretrained with 50 LLM-generated tasks, 136.10% for the model pretrained with 75 LLM-generated tasks, and 146.10% for the model pretrained with 150 LLM-generated tasks(Figure 8). It's important to note that, similar to the full Ravens simulation setup, the original training dataset and real-world demonstrations were included in the pretraining phase for the real-world experiments.

DISCUSSION

The collective results from both simulated and real-world evaluations provide strong evidence for the benefits of integrating LLM-generated tasks into training datasets for robotic manipulation models like CLIPort. By exposing the model to a wider array of complex tasks, CLIPort shows notable enhancements in performance across both familiar tasks and novel scenarios, evidencing an improved comprehension and adaptability.

The marked increase in zero-shot generalization capability suggests that LLM-generated tasks contribute to a training environment that effectively broadens the model's ability to extrapolate and apply learned behaviors in unfamiliar contexts. This is crucial for robots expected to function in diverse and unpredictable settings, where adaptability and generalization are key.

Furthermore, the significant performance improvements in real-world tasks underscore the practical implications of using LLMs to enrich training datasets. This advance not only enhances the immediate task performance but also opens

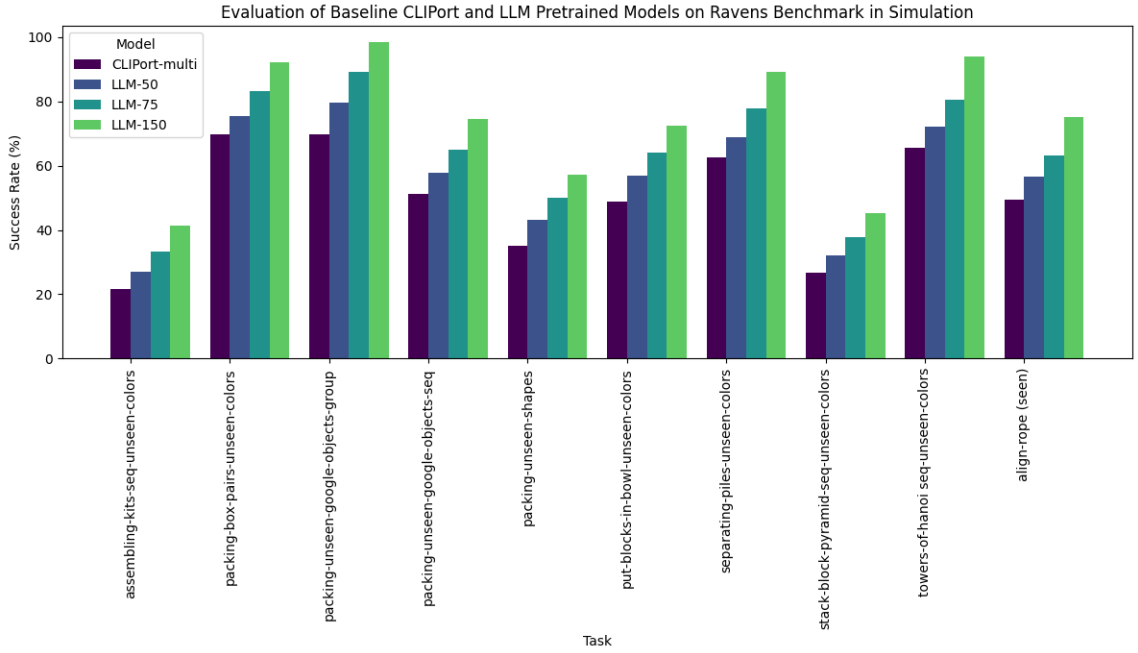

Fig. 6. Performance comparison of CLIPort and its pretrained variations on the full Ravens benchmark in simulation.

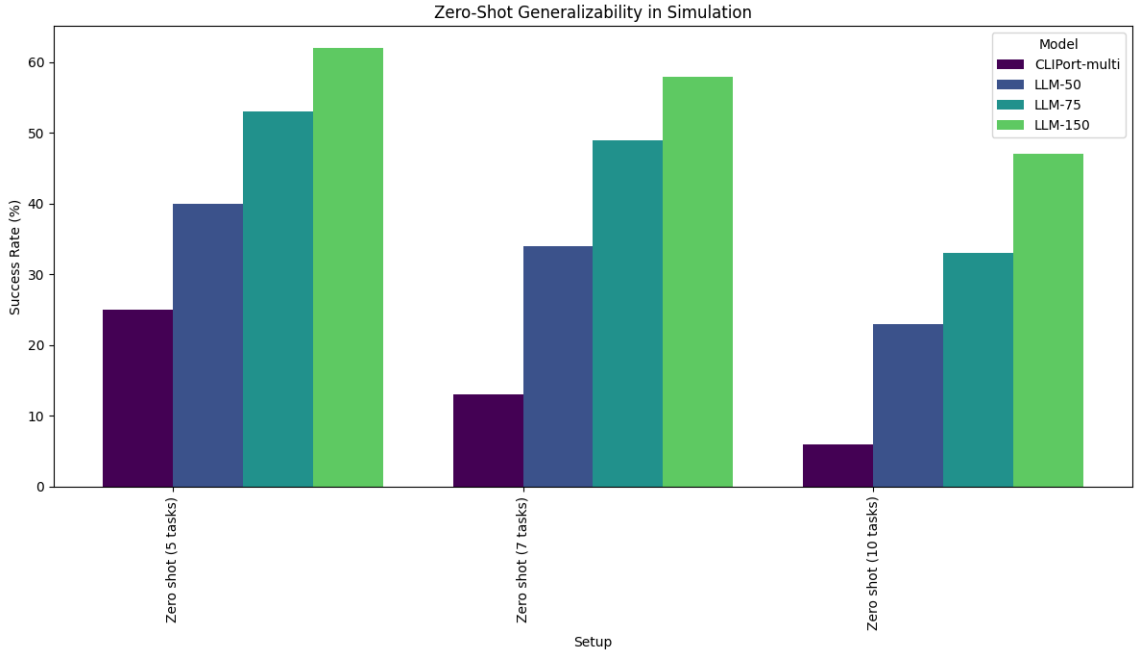

Fig. 7. Zero-shot generalization performance of CLIPort and its pretrained variations in simulation.

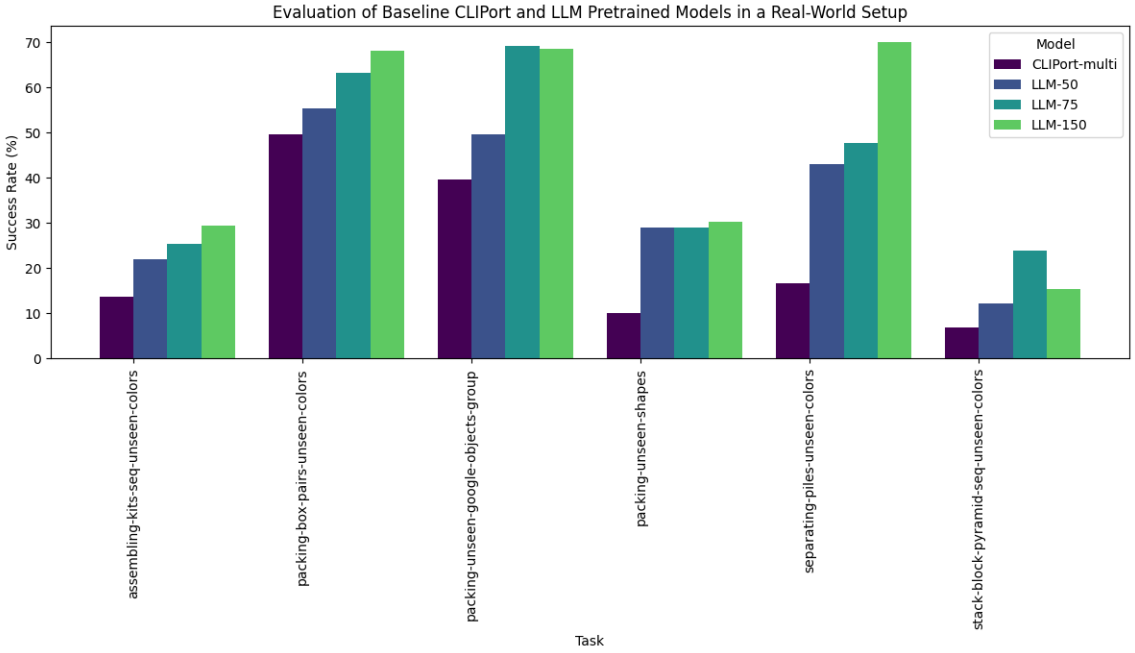

Fig. 8. Real-world performance comparison of CLIPort and its pretrained variations using the Xarm7.

pathways to broader applications of robotic systems beyond controlled laboratory environments.

Importantly, across all experiments, there is a direct correlation between the amount of LLM-task data incorporated into the training process and the observed improvements. Specifically, the comparison between models pretrained with 50, 75, and 150 LLM-generated tasks consistently demonstrates that greater diversity and volume of training data lead to better enhancements in the model's capabilities. This trend solidifies the argument for the strategic use of LLMs in creating enriched, diverse datasets for robotic training, paving the way for advancements in robotic manipulation that can adeptly handle a wide spectrum of real-world challenges.

Conclusion

This research has demonstrated the substantial benefits of integrating autonomously generated, LLM-augmented task datasets into the training of robotic manipulation models, specifically CLIPort. Through a comprehensive series of experiments, I have shown that the inclusion of diverse and complex tasks generated by CodeLlama-Instruct leads to marked improvements in model performance across various metrics and environments. My findings reveal that models pretrained on datasets augmented with 50, 75, and 150 LLM-generated tasks exhibit significant performance increases, with up to 52.99% in simulation tasks and up to 146.10% in real-world tasks with the XArm-7 robot equipped with a suction gripper, underscoring the effectiveness of my methodology.

The autonomous generation of task datasets represents a transformative step forward in the field of robotic manipulation. By significantly reducing the reliance on human labor for dataset creation, my approach not only streamlines the development process but also enhances the scalability and sustainability of model training. This methodology allows for the rapid and efficient production of highly diverse training data, which is crucial for the advancement of robotic systems capable of navigating and manipulating a wide range of environments with minimal human intervention.

Moreover, the remarkable increase in zero-shot generalization capability indicates that training with LLM-generated tasks equips models with a broader understanding and adaptability, enabling them to perform well in unfamiliar contexts. This quality is essential for the deployment of robots in real-world settings, where they must be able to handle tasks and scenarios that were not explicitly covered during training.

In conclusion, the use of LLMs to augment training datasets for robotic manipulation models represents a significant leap towards the development of more autonomous, versatile, and efficient robotic systems. As I continue to explore and refine this methodology, I anticipate further advancements in robotic manipulation research, paving the way for robots to adeptly navigate and manipulate their environments, thereby expanding their potential applications across diverse and unpredictable settings. My research underscores the pivotal role of autonomous dataset generation in achieving these advancements, highlighting its implications for the future of robotic systems in real-world applications.

References

[1] Brenna Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57:469–483, 05 2009.

[2] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation, 2018.

[3] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *CoRR*, abs/1504.00702, 2015.

[4] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[5] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation, 2021.

[6] Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Language-conditioned imitation learning for robot manipulation tasks, 2020.

[7] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Ayzaan Wahid, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation, 2022.