2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

Behavior-Actor: Behavioral Decomposition and Efficient-Training for Robotic Manipulation

Wenyi Jiang[1], Baowei Xv[2], and Zhihao Cui[2*]

***Abstract*— Language-conditioned Robotic manipulation demonstrates great potential in tackling various tasks. However, the generalization ability of this technique to unseen commands remains a challenge. Moreover, existing methods suffer from the burdensome overhead of data collection costs. Nowadays, Large language models (LLMs) have demonstrated impressive natural language understanding capabilities. In this work, we propose a novel scheme called Behavior-Actor(BehAct), which leverages the power of LLM to decompose language commands into executable behaviors in Retrieval-Augmented Generation(RAG) manner. A End-to-End actor is then trained to execute these identified behaviors. BehAct's LLM acts as a "brain", while the actor acts as a "hand". A single actor model is trained from scratch on 11 real-world tasks, 40 behaviors using 276 demonstrations, only 7 for each behavior in average. We achieve a 68% average success rate on seen commands, which aligns comparably with recent works. Moreover, BehAct exhibits an impressive 45% average success rate on unseen commands, doubling the performance of the baseline approach. In the BehAct system, LLM-agnostic design enables flexibility in leveraging advanced LLMs without necessitating fine-tuning. Our code has been made publicly available here.**

I. INTRODUCTION

Creating a robot capable of comprehending natural language commands within a physical environment presents an ongoing challenge. While robots excel in individual tasks[1], [2], integrating them into a real-world universal agent remains elusive. It is challenging to develop an multitask agent that can effectively generalize to previously unseen commands. Nowadays, Large Language Models(LLM)[3] brings new breath to the pathway to behavior understanding[4], [5], [6], [3], [7]. LLM exhibits the ability of decomposing the commands into behaviors. Our work concentrates on creating a real-world multitask robot empowered by LLM, capable of effectively interacting with previously unseen human commands.

While the aspiration of developing such a robot is compelling, its realization poses numerous challenges. The foremost obstacle lies in the extensive length of the Robotic manipulation pipeline, which entails multiple intricate stages such as perception, decision-making, planning, and control. In recent works[7], [8], [1], [9], [10], [11], end-to-end actor(policy) has shown substantial advantages. robots can effectively transfer knowledge across tasks.

This work was done when Wenyi Jiang was a intern in Mech-Mind.
[1]Wenyi Jiang is with Department of Computer Science, Geogria Institute of Technology, Shenzhen, China. wjiang311@gatech.edu
[2]Baowei Xv and Zhihao Cui are with the Deep Learning Group, Mech-Mind, Beijing, China. cui.zhihao@mech-mind.net;xu.baowei@mech-mind.net
[3]*Corresponding author

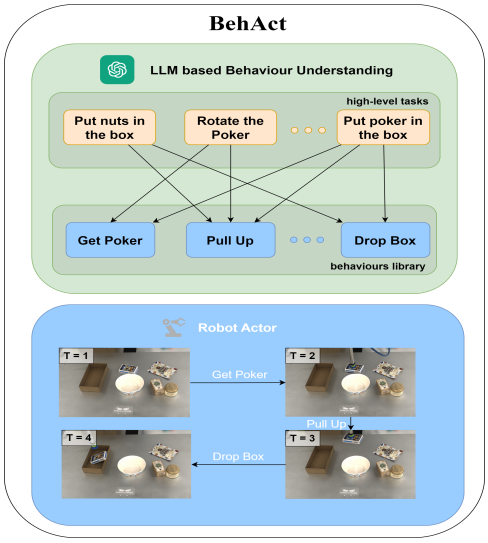

Fig. 1. The overview of BehAct architecture. High-level commands are usually supported by various subbehaviors (primitives) The LLM understand the breakdown rules by background information searched using RAG. The actor learns how to perform the behaviors by behavior cloning. During deployment, the LLM decomposes high-level tasks into sub-behaviors, which exist in a behavior library. Subsequently, the Robot Actor sequentially performs these sub-behaviors.

Another primary challenge in Robotic manipulation lies in the robot's struggle to comprehend various language commands effectively. Users often present commands in paraphrased forms, wherein expressions like "I would like to have some fruits" might be alternatively phrased as "Are there some fruit I can eat?". The limitation in behavior comprehension hinders the robot's ability to learn the underneath relationship between tasks. For instance, even if an agent has learned to "put the poker in the bowl" and "put the nuts in the box," it may still encounter difficulty comprehending the command "put the poker in the box". This inadequacy in understanding poses a hurdle in enabling the robot to perform optimally across a range of unseen tasks with a limited number of demonstrations.

To tackle these challenges, we present a novel scheme of Behavior-Actor abbreviated as **BehAct** for language-conditioned robotic manipulation. Leveraging the capabilities of a Large Language Model (LLM), specifically GPT-4[12] (Generative Pre-trained Transformer), we effectively segment the user's command into distinct behavioral steps which an actor model is able to handle. The LLM is robust to not only simple augmentation for text prompts, but context-sensitive or deeply paraphrased commands.

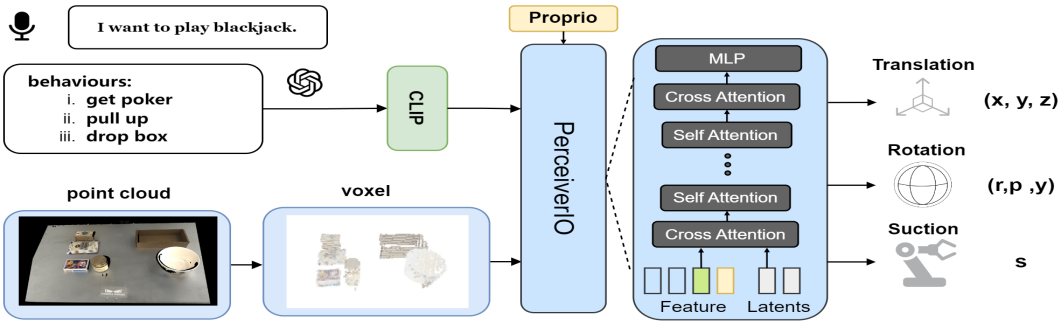

Fig. 2. The pipeline of BehAct. BehAct is a action predict model trained with imitation learning. Initially, the voice commands are transcribed into text format. This textual input is then processed by the Large Language Model (LLM) to generate corresponding behaviors. BehAct operates within an observe-act loop, continually executing actions until all behaviors are completed. The colorful point cloud is voxelized, combined with the language embedding extracted by CLIP. The PerceiverIO backbone use self-attention and cross-attention to fuse the two modalities. The backbone output the Q-value for translation, rotation and suction state. Classifier heads are employed to generate actions for execution.

Our end-to-end actor establishes a seamless connection between visual and linguistic(V+L) information and the 6-DoF pose of robot arm. Operating with an action-centric paradigm, the actor eliminates the need for explicit object detection and gripper pose generation. Being aware of the behaviors rather than tasks, our actor needs merely about 10 demonstrations to perform steadily. Additionally, we employ CLIP[13]to extract meaningful semantic embeddings from the behavioral prompt. By integrating the attention mechanism of the Transformer architecture, we skillfully fuse the visual and language modalities.

Our main contributions can be summarized as follows:

1) We introduce a new scheme for robotic manipulation called BehAct. By leveraging LLM for behavior understanding in RAG style, BehAct exhibits generalization to unseen commands and remains resilient to paraphrased instructions. We outperformed recent works by 68% average success rate on seen commands and 45% on unseen commands.
2) Behavior understanding also speed up the training for multitask actor. Being aware of the same behaviors between tasks, our actor waive the need for redundant data collection and costly V+L model retraining. For the unsteady behaviors, we adopt HG-DAgger[14] style to supplement behavior demonstrations instead of re-collecting the entire task.
3) The BehAct scheme demonstrates adaptability for future enhancements, remaining agnostic to specific Large Language Models (LLMs) and capable of leveraging the enhanced capabilities offered by forthcoming LLM iterations.

II. Related Work

A. Language Understanding for Robotics

Language understanding has been studied extensively in the field of robotics. Prior works [15], [16] have traditionally utilized tools like lexical analysis, formal logic, and graphical models to interpret language instructions. However, contemporary approaches leveraging Large Language Models (LLMs) are emerging with significant potential. Instruct2ACT [4], CaP [5], and VoxPoser [6] employ LLMs to generate Python code for robotic control using primitives. Meanwhile, PaLM-E and SayCan [3], [7] guide robots through complex, long-horizon manipulation tasks. RT-2 [17] leverages internet-scale data and fine-tunes a Vision-and-Language Model (VLM) to generate executable robot actions. In general, the utilization of LLMs in previous work can be grouped into three categories: prompt engineering, fine-tuning, and retraining. We elaborate the advantages of our Retrieval-Augmented Generation(RAG) style in discussion sections.

Similar to PaLM-E[3] and SayCan[7], our work breaks down high-level commands into behaviors that the actor can perform. In contrast, instead of training the LLM to generate the probability of sub-behaviors, we facilitate its understanding of the scene and examples through information retrieval systems. This involves retrieving relevant information from external sources and using it to enhance the generation process.

B. Language-Conditioned Robotic Manipulation

There is a long history of exploring the bridge between language and behavior. A lot of prior works[18], [1], [2] utilize imitation learning, reinforcement learning to fill the gap. Some works[4], [19] harness the pre-trained foundation models like SAM[20], CLIP[13] to perform zero-shot generalization. While pretrained models significantly enhance generalization abilities, they often pose challenges when fine-tuning on self-possessed datasets, leading to limited scalability. Conversely, our approach adopts few-shot learning, facilitating straightforward incorporation of new behaviors for the LLM to deconstruct. This distinction highlights the greater adaptability and flexibility of our methodology. The most relevant works to our project is the end-to-end approaches[8], [1]. The End-to-End architecture enhances scalability and simplifies design. However, existing approaches lack robustness against command paraphrasing and a comprehensive understanding of semantic meanings. While excelling at individual tasks, these approaches often overlook shared underlying behaviors beneath high-level tasks. In contrast, our approach focuses on training an end-to-end model with

common behavioral primitives applicable across tasks. This strategy enables the robot to generalize to unseen commands and withstand paraphrasing challenges effectively.

III. Approach

A. Problem Formulation

BehAct is trained using imitation learning, which involves utilizing observation-action pairs $\mathcal{D} = \{d_1, d_2, ..., dn\}$. A demonstration d contains an observation o and an action a

$$d = (o, a) \tag{1}$$

where observation $o = (\mathcal{I}, \mathcal{L}, s)$ is the combination of RGB-D Image $\mathcal{I}$, $\mathcal{I} \in \mathbb{R}^{H \times W \times 4}$, behavior instruction $\mathcal{L}$ (e.g., "get blue block") and robot proprioception s. It is important to note that a high-level command list like "I am thirsty" can entail ambiguity. $\mathcal{L}$ is the sub-behavior which is breakdown from the high-level command by the LLM. And $s \in \{0, 1\}$ is the suction cup's open state at present. The action a encompasses the expert operation's 6-DoF pose and suction cup's open state.

Q-value based Imitation Learning we adopt the Deep Q-learning to learn the best action a given an observation o. Our robotic system can be formulated by $\Phi_\epsilon(a|o)$, parameterized by ϵ , which maps the observation to a predicted action $\tilde{a} = \Phi_\epsilon(o)$. Our optimization goal is minimizing the distribution difference $\mathcal{M}$ of predicted action $\tilde{a}$ and the expert's action a in the dataset $\mathcal{D}$.

$$\epsilon^* = \arg\min_{\epsilon} \mathbb{E}_{(s,o)\sim\mathcal{D}} \left[\mathcal{M}\left(\Phi_\epsilon(o), a\right)\right] \tag{2}$$

We use PerceiverIO[21] model to estimate the Q-value $Q(o, a)$ of action an under observation o . The predicted action is simply the action with maximum Q-value $\tilde{a} = \arg\max_a Q(o, a)$.

B. LLM based behavior understanding

LLM(Large Language model) is responsible for understanding the high-level task. The given high-level language command $\mathcal{C}$ will be split by the LLM into several behaviors, which exist in the executable behavior library $\mathcal{S}$. i.e. $\mathcal{C} \rightarrow [\mathcal{L}_1, \mathcal{L}_2, ..., \mathcal{L}_m], \mathcal{L}_i \in \mathcal{S}$, e.g., "I want to play blackjack." can be understood and decoupled into following 3 behaviors, ["get poker", "pull up", "drop box"]. All behaviors in the library $\mathcal{S}$ has been trained on an actor model through behavior cloning. Then we use an observation-act loop to execute the actor m times to execute the task. Each time, i we give the actor model $\mathcal{L}_i$ as the language prompt for the actor model.

Retrieval-Augmented Generation (RAG) In all experiments, we employ GPT-4[22] as our Large Language Model (LLM). The LLM is enriched with scene information to aid its understanding, encompassing workstation settings and relative positions of the robotic arm, camera, and table. To facilitate the learning process, we manually create a comprehensive behaviors list $\mathcal{S}$ with its effects and restrictions in our settings. Moreover, we present examples of breakdown instructions provided by experts. All of this background information is saved into a knowledge source. With a user's query, the RAG system will search and supply the most relevant information based on the similarity score of embeddings. The embeddings of both the query and the information are extracted through ERNIE 4.0. To maintain control over the generated behaviors, we limit the LLM to produce no more than 10 behaviors. Some example of background information used for the RAG can be found here.

C. Actor Behavior Cloning

Figure 2 demonstrates the robot actor's pipeline employed in our approach. For the actor's backbone, we follow the implementation of PerAct[8]. The RGB-D image $\mathcal{I}$ is voxelized. Initial down-sampling occurs through 3D convolution and max-pooling. The proprioceptive information s is tiled in 3D to match the voxel's dimensions. CLIP's text encoder is utilized to generate the language embedding for the behavior prompt, which is subsequently concatenated with the voxel. To recover positional information loss in the Transformer, positional embedding is appended to the tensor. The fusion of language embedding and voxel representation is performed using the PerceiverIO[21] model. Finally, the Q-function is employed to predict the 6-DoF pose.

Voxelization We voxelize the RGB-D image with the voxel grid size of 100^3, $\mathcal{I} \Rightarrow v, v \in \mathbb{R}^{100^3}$. We manually set the scene bounds to a Region of Interest (ROI) corresponding to 0.7m*1.0m*0.3m, which includes all objects but excludes the table surface. The 6-DoF poses are discretized to align with the nearest grid point and a discrete rotation class, with a rotation resolution of 5 degrees. The suction state is represented as a binary value indicating the open or closed state of the suction cup.

CLIP In our approach, we employ the text encoder from CLIP to convert the behavior prompt into a semantic embedding($\mathcal{L} \Rightarrow \mathcal{C}, \mathcal{C} \in \mathbb{R}^{77\times512}$), enabling effective fusion of language information with visual representations.

PerceiverIO The PerceiverIO model[21] $\mathcal{P}$ serves for handling data from arbitrary modality. Leveraging both self-attention and cross-attention mechanisms, the PerceiverIO model efficiently integrates inputs from visual and linguistic modalities $\mathcal{F} = \mathcal{P}(\mathcal{C}, v)$. In our approach, the voxel v is divided into patches and fed into the PerceiverIO model. Subsequently, the output embeddings are upsampled to the original voxel size.

Q-Functions To translate the per-feature $\mathcal{F}$ of PerceiverIO into actionable outcomes $\tilde{a}$, we employ Q-learning. Utilizing three separate classifier heads, predict the discrete tool center point (TCP) Euclidean position, rotation angles, and suction state. Classifier heads employ Multilayer Perceptrons (MLPs) to decode the Q-value $Q(a|\mathcal{F})$. Specifically, "Translation" indicates the voxel grid index where the TCP is positioned, "Rotation" signifies the three Euler angles, and "Suction" involves a binary value determining the status of the suction (open or closed). The class with the highest Q-value for each classifier head serves as the final prediction, subsequently

executed by a motion planner.

$$\begin{aligned} \tilde{a} &= (trans, rot, suction) \\ trans &= \arg\max_{(x,y,z)} Q((x,y,z)|\mathcal{F}), \quad x,y,z \in \mathbb{N}^{100} \\ rot &= \arg\max_{(r,p,y)} Q((r,p,y)|\mathcal{F}), \quad r,p,y \in \mathbb{N}^{360/res} \\ suction &= \arg\max_{s} Q(s|\mathcal{F}), \quad s \in 0,1 \end{aligned} \tag{3}$$

Here, x, y, and z represent the voxel coordinates, while r, p, and y correspond to the roll, pitch, and yaw angles of the TCP. The rotation resolution is denoted as res, which is set at 5 degrees. s signifies the suction cup's open state.

IV. Experiment

A. Loss

We trained BehAct with the cross-entropy loss as follows:

$$\begin{aligned} \mathcal{L}_{entropy} = &-Y_{trans} log(\sigma(Q(trans|o))) \\ &-Y_{rot} log(\sigma(Q(rot|o))) \\ &-Y_{suction} log(\sigma(Q(suction|o))) \end{aligned} \tag{4}$$

where σ is the Softmax function. o is the observation. $Y_{trans}, Y_{rot}, Y_{suction}$ is the one-hot label correspondingly.

B. Real-World Environment

Dataset Collection A total of 320 demonstrations were gathered through a human-robot interaction loop, with 44 demos for validation and 276 for training purposes. We have 40 behaviors in total. In average, a single behavior only have 7 demonstrations for training. The initial dataset batch comprised entirely experts' behaviors. After the first version of the model was trained, we adopted a HG-DAgger[14] style approach for data collection. In the observation-action loop, we allow the robot to act following the predicted action, under the supervision of an expert. If an action was deemed unlikely to achieve the behavior goal, the loop is temporarily interrupted and the action was substituted with an expert's action. This strategy allowed us to streamline data collection efforts, focusing on challenging scenarios and mitigating bias. Notably, the most common issue our model encountered was incorrect object selection. To tackle this, we deliberately incorporated identical scenes with varying object prompts, like "get poker" versus "get oil," while keeping the visual observations consistent. This approach effectively encourages the model to glean insights from language prompts rather than relying mainly on vision observation.

C. Evaluation Metric

To evaluate the performance of our model. We report the average success rate on 11 tasks (22 seen variations and 20 unseen variations). A task is decided to be successful if the language commands is fulfilled, which is judged by expert. No partial credit is counted.

We group the test set into two classes, seen and unseen tasks. For seen tasks, the model will receive exactly the same commands as training to execute, but with different arrangement of objects. For unseen tasks, we manually select executable commands but unseen commands for model to execute. Specifically, we replace the object in the commands with objects appeared in other training tasks. The detail of tasks can be found at here.

V. Results

Baseline Methods

C2FARM-BC Coarse-to-fine-grain (C2FARM) [23], introduced by James et al., stands as a state-of-the-art robot manipulation approach assessed on the RLBench dataset. Functioning as a fully-convolutional network, it takes any-dimensional 3D voxel as input and yields corresponding actions as output. Its notable trait lies in its multi-level voxelization. In our experimentation, we configured the voxel size to 32^3 and the grid's dimensions to 0.1^3m, in line with James et al.'s setup. Additionally, the language embedding derived from CLIP is incorporated into the voxel embedding.

PerAct Perceiver-Actor [8] represents a closely related work to our approach. It expands upon CLIPort [2], extending its capabilities into the 3D domain and delivering an end-to-end solution for robot manipulation. Similar to our methodology, it employs the Transformer architecture to merge visual and linguistic observations. Consistent with Shridhar et al., we configure Perceiver-Actor's latent dimension to 512. The voxel size is set at 100^3, corresponding to a 1^3m space, while rotation resolution remains at 5 degrees.

A. multitask Performance

Table I offers a performance comparison among C2FARM-BC, PerAct, and our BehAct. C2FARM-BC exhibits relatively limited performance, attributed to its restricted perceptual field. This fully-convolutional architecture struggles to establish long-range relationships, particularly in shallower layers. Utilizing a similar perceiver backbone as PerAct, our model shows comparable results on seen tasks.

On unseen tasks, both C2FARM-BC and PerAct's performance drops significantly. The C2FARM architecture wrestles with interpreting linguistic commands and often replicates trained skills in unseen scenarios. PerAct demonstrates a 23% average success rate (ASR) on unseen scenes, suggesting that while it may learn from language prompts, its learning process falls short. PerAct encounters challenges in acquiring a nuanced comprehension of object and action concepts. In contrast, empowered by LLM, BehAct effectively disentangles commands into actions and objects. BehAct attains an average success rate of about 45%, double that of PerAct. The pivotal role of behavior understanding in enhancing interpretability and generalization is evident.

Practically, PerAct often errs in selecting objects for unseen commands, drastically undermining its generalization ability. With a comprehensive behavior library, BehAct gains a nuanced understanding of diverse tasks like "picking", "placing", and "pushing".

We further assess the model's performance across training iterations. Figure 3 illustrates the average success rate (ASR) trends for the Put In task. Our BehAct is trained on all 11 tasks, and at 120k iterations, it exhibits a 20% higher ASR

TABLE I

COMPARISON RESULTS FOR DIFFERENT METHODS TESTED WITH SEEN OR UNSEEN LANGUAGE COMMANDS

	Put in		**Move**		**Stack**		**Cover**		**Uncover**		**Close**	
Success Rate(%)	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**
C2FARM-BC	20	0	50	0	15	10	25	0	15	15	0	N/A
PerAct	35	0	**100**	0	**40**	5	**40**	0	70	45	**30**	N/A
BehAct	**55**	**35**	**100**	**90**	35	**35**	**40**	0	**80**	**55**	**30**	N/A
	Hit		**Rotate**		**Bring**		**Push**		**Press**		**Average**	
Success Rate(%)	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**	**Seen**	**Unseen**
C2FARM-BC	45	10	0	0	65	5	0	0	**100**	N/A	30	5
PerAct	85	0	65	0	**90**	10	70	60	90	N/A	65	23
BehAct	**100**	**45**	**70**	**30**	85	**70**	**80**	**65**	95	N/A	**68**	**45**

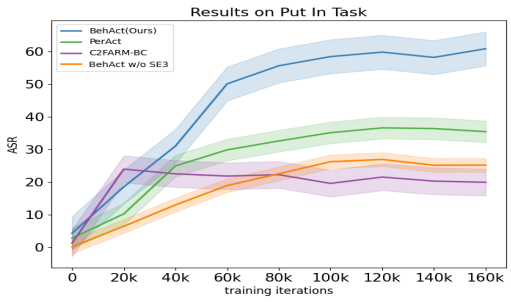

Fig. 3. The line chart of average success rate(ASR) On Put In task With Seen Commands.

than PerAct. This improvement underscores BehAct's ability to benefit from behavior understanding, as the behavior prompt diminishes ambiguity in high-level tasks. In comparison to C2FARM-BC, both PerAct and BehAct outperform due to their Transformer architecture.

B. Ablation Studies

TABLE II

ABLATION STUDIES OF EFFECTIVENESS OF SE3, LANGUAGE COMMANDS AND MORE DATA

Average Success Rate On 'put in' task	Single-task		multitask	
	Seen	**Unseen**	**Seen**	**Unseen**
BehAct	65	0	55	35
BehAct w/o SE3	10	0	25	20
BehAct w/o Lang	50	0	45	0
BehAct w/ 20 demos	15	0	N/A	N/A

In this chapter, we aim to address the following four research questions:

1) Does multitask training provide benefits for single-task performance?
2) How effective is the SE3 augmentation technique?
3) What role do language prompts play in the language-conditioned robotic manipulation process?
4) Does the scalability of the system improve with an increase in the number of demonstrations?

We present two sets of model results in this study. The first set consists of a model trained solely on the "put in" task, while the second set includes a model trained on all 11 tasks. For each model, we explore different training settings, including the presence or absence of SE3 augmentation, the absence of any language commands, and training with only 20 demonstrations compared to the total of 80 demonstrations. We evaluate the performance of these models on a single task, "putting <item>in <container>". This task has two variations, resulting in a total of 20 test instances per model.

As shown in Table 2, BehAct does not consistently benefit from the inclusion of other tasks in the training process. The multitask model exhibits a 10% drop in ASR compared to the single-task model. However, we observe potential benefits for other tasks through multitask learning. The joint training of multiple tasks can lead to both inaccurate actions and a better understanding of objects and their associated behaviors, which leads to task-specific consequence. We observe that certain behaviors, such as "pull up", are shared across different tasks. The model can enhance its understanding of these concepts by studying the behaviors of other tasks. Additionally, BehAct is capable of learning the concept of other objects through exposure to different tasks. For example, it can learn what an "oil" is by observing the "rotate oil" variation.

The inclusion of SE3 augmentation is crucial for enhancing the effectiveness of the demonstration utilization. Without SE3 augmentation, the average success rate significantly decreases from 65% to 10% for the single-task model and from 55% to 25% for the multitask model. The absence of SE3 augmentation in the training process results in a lack of generalization ability, causing the model to overfit to specific object arrangement settings. By incorporating SE3 augmentation, the model gains the capability to handle various layouts of objects and mitigates the sparsity of rewards in the search space.

We also investigate the influence of the Language goal on BehAct. For comparative analysis, we provide empty text inputs. Intriguingly, both the single-task model and the multitask model exhibits meaningful behavior even in the absence of explicit language cues. Due to the Q-Learning, BehAct always predict the most possible action given a perception. The vision information only is enough for it to do a single variation behavior cloning task, like "press the button". However, it will become aimless when we are expecting it to perform an unseen task. Hence, language prompt is the key to distinguish between variations and unseen commands understanding.

VI. DISCUSSSIONS

A. LIMITATAIONS

While BehAct demonstrates the ability to perform complex tasks, the limited number of demonstrations makes it sensitive to variations in shape, color, and lighting conditions. A common error encountered in practice is that of incorrect object picking. For instance, when learning with a language prompt like "pick up the banana," the model might not form the concept of a banana. Instead, it might simply learn to pick the object on the left. To address these generalization challenges, data collection in an HG-DAgger manner and SE(3) augmentation proved helpful. Additionally, we expect that simulated environments may offer further advantages.

B. Comparison with Other Methods

There are two main advantages of our work compared to other methods using LLMs: scalability and effectiveness. In practice, users, especially those in industries, often need to customize their own tasks rather than perform pre-trained tasks on an agent. However, they typically lack the resources for fine-tuning or retraining the model. Additionally, collecting paired data of behavior and diverse language prompts is expensive and burdensome. Lastly, the LLMs are also well known for its hallucination and catastrophic forgetting problems. Thus, compared to methods that use fine-tuning or retraining for Vision-Language Models [24], [4], [6], [7], our approach offers the advantage of allowing agents to learn new behaviors with just a few new demonstrations (an average of 7) and some background information in RAG. By decoupling behavior decomposition and behavior learning, we eliminate the need for extensive data collection and costly Vision-Language model retraining.

On the other hand, compared to prompt engineering, RAG efficiently retrieves and incorporates only the most relevant information from a vast knowledge base at the time of generation, making it computationally efficient. RAG also scales effectively by maintaining a large, searchable knowledge base and retrieving relevant information as needed without increasing the prompt length.

VII. CONCLUSIONS

In conclusion, our novel robotic manipulation scheme, BehAct, adeptly connects perception (V+L) and action through the combination of LLM and Transformer. Its robustness to paraphrase and interpretability are notable features. The end-to-end training pipeline simplifies design and supports multitask learning. Real-world experiments demonstrate BehAct's impressive ability to effectively generalize to unseen language commands.

REFERENCES

[1] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "Bc-z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 991–1002.

[2] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2022, pp. 894–906.

[3] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[4] S. Huang, Z. Jiang, H. Dong, Y. Qiao, P. Gao, and H. Li, "Instruct2act: Mapping multi-modality instructions to robotic actions with large language model," *arXiv preprint arXiv:2305.11176*, 2023.

[5] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 9493–9500.

[6] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[7] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[8] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 785–799.

[9] C. Lynch and P. Sermanet, "Grounding language in play," *arXiv preprint arXiv:2005.07648*, vol. 3, 2020.

[10] S. Nair, E. Mitchell, K. Chen, S. Savarese, C. Finn *et al.*, "Learning language-conditioned robot behavior from offline data and crowd-sourced annotation," in *Conference on Robot Learning*. PMLR, 2022, pp. 1303–1315.

[11] O. Mees, L. Hermann, and W. Burgard, "What matters in language conditioned robotic imitation learning over unstructured data," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 11 205–11 212, 2022.

[12] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

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

[14] M. Kelly, C. Sidrane, K. Driggs-Campbell, and M. J. Kochenderfer, "Hg-dagger: Interactive imitation learning with human experts," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8077–8083.

[15] Y. Jiang, S. S. Gu, K. P. Murphy, and C. Finn, "Language as an abstraction for hierarchical deep reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[16] D. I. A. Team, J. Abramson, A. Ahuja, A. Brussee, F. Carnevale, M. Cassin, F. Fischer, P. Georgiev, A. Goldin, M. Gupta *et al.*, "Creating multimodal interactive agents with imitation and self-supervised learning," *arXiv preprint arXiv:2112.03763*, 2021.

[17] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[18] C. Lynch and P. Sermanet, "Language conditioned imitation learning over unstructured data," *arXiv preprint arXiv:2005.07648*, 2020.

[19] Y. Mu, Q. Zhang, M. Hu, W. Wang, M. Ding, J. Jin, B. Wang, J. Dai, Y. Qiao, and P. Luo, "Embodiedgpt: Vision-language pre-training via embodied chain of thought," *arXiv preprint arXiv:2305.15021*, 2023.

[20] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," *arXiv preprint arXiv:2304.02643*, 2023.

[21] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer *et al.*, "Perceiver io: A general architecture for structured inputs & outputs," *arXiv preprint arXiv:2107.14795*, 2021.

[22] OpenAI, "Gpt-4 technical report," 2023.

[23] S. James, K. Wada, T. Laidlow, and A. J. Davison, "Coarse-to-fine q-attention: Efficient learning for visual robotic manipulation via discretisation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 13 739–13 748.

[24] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.