2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Conditionally Combining Robot Skills using Large Language Models

K.R. Zentner[*]
Univ. of Southern California
kzentner@usc.edu

Ryan Julian
Google DeepMind
rjulian@google.com

Brian Ichter
Google DeepMind
ichter@google.com

Gaurav S. Sukhatme[+]
Univ. of Southern California
gaurav@usc.edu

***Abstract*— This paper combines two contributions. First, we introduce an extension of the Meta-World benchmark, which we call "Language-World," which allows a large language model to operate in a simulated robotic environment using semi-structured natural language queries and scripted skills described using natural language. By using the same set of tasks as Meta-World, Language-World results can be easily compared to Meta-World results, allowing for a point of comparison between recent methods using Large Language Models (LLMs) and those using Deep Reinforcement Learning. Second, we introduce a method we call Plan Conditioned Behavioral Cloning (PCBC), that allows finetuning the behavior of high-level plans using end-to-end demonstrations. Using Language-World, we show that PCBC is able to achieve strong performance in a variety of few-shot regimes, often achieving task generalization with as little as a single demonstration. We have made Language-World available as open-source software at https://github.com/krzentner/language-world/.**

I. Introduction

Combining skills to perform new tasks is an area of active research in machine learning and robotics. Skill reuse has several potential advantages, such as allowing a robot to efficiently re-use data from old tasks to rapidly learn new tasks. Sequencing skills might also help address the difficulty of learning long-horizon tasks end-to-end. Here, we propose one way of using text to combine skills that leverages the capabilities of recent large language models (LLMs).

Our approach uses a textual description of how to perform a task we term a "conditional plan." A conditional plan lists a set of skills to use to perform a task and associates a linguistic condition under which each skill should be active. See Table I for an example. Because the plan expresses conditional behavior, our method is able to make use of an LLM without querying it while the robot is performing a task. Instead, a much simpler neural architecture with several orders of magnitude fewer parameters implements the conditional behavior. This maintains the runtime efficiency of other end-to-end approaches and allows fine-grained skills. Because the conditional plan contains natural language, it can also improve interpretability and transparency by describing what skill the robot is using and why it is using it. Further, it provides additional control compared to other approaches using LLMs, since a human can inspect or edit the conditional plan before it is used.

[*]Continuation of work done as an intern at Google Brain
[+]GSS holds concurrent appointments as a Professor at USC and as an Amazon Scholar. This paper describes work performed at USC and is not associated with Amazon.

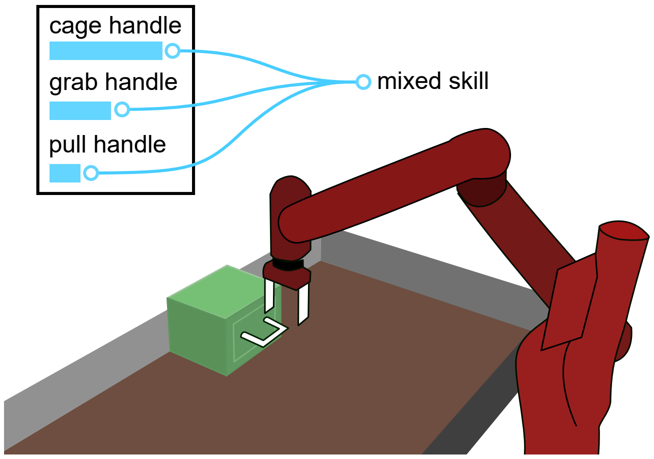

Fig. 1: This figure shows a simulated robot using our method. See Figure 5 for a detailed description of how it functions.

In Section III we extend the popular "Meta-World" benchmark to create "Language-World." Language-World includes a set of tools for working with language that allow us to evaluate our proposed method and compare it to several ablations, including using an LLM directly.

In Section IV, we describe conditional plans and how we use an LLM to generate them. Then, we propose a method called Plan Conditioned Behavioral Cloning to finetune the behavior of plans using end-to-end demonstrations. Finally, we describe how to perform effective few-shot generalization using Cross-Task Co-Learning, including using PCBC.

The primary contributions of this work are as follows:

1) A new benchmark, "Language-World," which extends the Meta-World benchmark to make performing experiments with large language models on it practical.
2) A method, Plan Conditioned Behavioral Cloning (PCBC), that allows finetuning the behavior of high-level plans using end-to-end demonstrations.
3) Experimental demonstrations that PCBC and Co-Learning are able to achieve strong few-shot generalization results in "Language-World."

II. Related Work

a) Large Language Models: Recent work in language modeling has resulted in "large language models" (LLMs), which contain billions of neural network parameters and demonstrate powerful zero and few-shot reasoning capabilities. In this work, we experiment with using three such LLMs: GPT-3 [5], GPT-3.5 [2], and PaLM 2 [6], [1]. This is an area of active research, and additional LLMs have become available since we began this work, including GPT-4 [23], LLaMa [32], and Claude [4]. Several methods to improve the utility of LLM output for downstream tasks have also been proposed, including finetuning with RL [24], finetuning with supervised learning [28] or improved prompting [35], [33], [40]. In this work, we experiment with a variety of prompt formats that make use of chain-of-thought [35], which is often able to improve the quality of LLM output with minimal effort.

b) Deep Reinforcement Learning (RL), End-to-End (E2E) Learning for Robotics: Learning robotic capabilities via RL has been studied for decades [14], [20], [19], [31]. More recent advances in neural networks that allow feature learning and fitting to complex high-dimensional functions have allowed end-to-end training of neural policies using RL [8], [22] and imitation learning (IL) [39], [7]. A number of simulated environments for benchmarking these end-to-end methods on robotic tasks exist, including Meta-World [38], which we extend in this paper.

c) Skills, Options, and Hierarchy in E2E Learning: Learning reusable skill libraries is a classic approach [9] for efficient acquisition and transfer of robot motion policies. Prior to the popularity of E2E methods, several methods [25], [29], [41] were proposed for acquiring a set of reusable skills for robotic manipulation. More recent E2E methods have been proposed for learning manipulation skills [37], [36], [15] as well as skill decomposition methods for learning and adaptation in locomotion and whole-body humanoid control [26], [10], [21], [17], [11], [13]. Although these methods have demonstrated some improvements to the sample efficiency of RL, significant improvements in complex environments remain elusive.

d) Large Language Models as Agents: Several recent works attempt to produce agents with useful zero-shot behavior by leveraging the generalization capabilities of large language models while mitigating their weaknesses at multi-step reasoning. Some approaches, such as [3], [30], [34], use an LLM to choose from a set of high-level actions described with natural language. Other approaches, such as [18], [27], use an LLM to generate code which is then executed to produce behavior. The method proposed in this paper exists in a middle ground between these approaches, where an LLM is used to generate code in a particular format that allows actions to be described with natural language, and the behavior of the program (i.e. conditional plan) can be tuned E2E.

III. Language-World

The Meta-World benchmark has emerged as a popular simulated environment for evaluating machine learning methods in the context of robotics, receiving over 150 citations in the past year alone. It provides 50 robotic manipulation tasks with a continuous range of random goals available for each task. The last year has also seen a rapid increase in interest with using large language models (LLMs) for robotics. Ideally, research using LLMs for robotics would use benchmarks that allow quantitative comparisons to methods that do not use LLMs. Language-World makes it relatively easy to perform these comparisons by providing three items that are useful, or necessary when using LLMs with Meta-World.

First, Language-World provides a brief natural-language description of each task e.g. "push the button on the coffee machine." Second, Language-World provides a query answering function (QAF), which allows the evaluation of a boolean semi-structured language query about a Meta-World state. This module performs similarly to a VQA model optimized for Meta-World, while avoiding the overhead of rendering images. By using the query answering function, methods that use language but do not deeply integrate visual processing with language can be efficiently evaluated on Meta-World tasks. The third item Language-World provides is a set of 30 scripted skills. These scripted skills can be used to reliably perform all tasks in the MT10 task set but can be applied to any of the 50 tasks. However, note that the method we propose in this work does not use the scripted skills. These three items allow Language-World to be used to perform simulated robotics experiments that can be easily compared to existing results in the literature.

a) ***Task Descriptions****:* Each of the 50 tasks in MT50-language has a one-sentence natural language description of the task. These descriptions can be used in a variety of ways, such as for conditioning a multi-task policy, or as inputs to an LLM. In our experiments below, we use these descriptions as conditioning in a baseline method (Descriptor Conditioned BC), as well as to prompt an LLM to generate a plan in Plan Conditioned BC.

b) ***Query Answering Function****:* The Language-World query answering function (QAF) is able to evaluate semi-structured textual queries on a Meta-World state. These queries resemble natural language e.g. "the gripper is around the puck," and the QAF answers each of them with a boolean value. The QAF can evaluate 13 simple geometric relationships between all objects in the task, such as "near," "left of," "in front of," or "below." The QAF can evaluate these relationships between all objects in all 50 tasks, including the robot's gripper as well as objects not present in the observation but at fixed locations, such as walls. It also can evaluate other useful cases, such as whether an object is touching the table, or if the robot's gripper is closed. Finally, this function can handle negation, simple conjunctions of the above, and perform basic inference to identify repeated subjects, allowing evaluating conditions like "Is the gripper open and not above the puck". The QAF also provides a list of all supported queries, so queries outside of the supported set

Condition	Skill
the gripper is closed and not near the drawer handle	open the gripper
the gripper is not near the drawer handle	move the gripper above the drawer handle
the gripper above the drawer handle	move the gripper down around the drawer handle
the gripper is open and around the drawer	close the gripper
the gripper is closed and around the drawer	pull the drawer open

TABLE I: Conditional Plan for `drawer-open`

can be mapped to the nearest supported query using a sentence embedding or using string edit distance. For simplicity of interpretation in this work, we use string edit distance when necessary.

*c) **Scripted Skills**:* Language-World provides 30 scripted skills, each of which has a brief natural language description (e.g. "put the gripper above the drawer handle"). These scripted skills have been tested using a hand-crafted mapping between queries and skills, and are able to perform all of the tasks in MT10-language with a success rate of over 90%. These scripted skills are stateless linear controllers extracted from the Meta-World scripted policies. Because most tasks require use of more than one skill, there are more skills than tasks, even though some tasks share skills. We use these scripted skills to compare the performance of 3 LLMs in Figure 2 and in evaluating different plan formats in Figure 4. However, we **do not use scripted skills in our remaining experiments.**

*d) **Task Formalism**:* Language-World uses the description of a task that is frequently used when performing multi-task RL with Meta-World. Specifically, each of the 50 available named tasks τ is an infinite horizon fully observable Markov Decision Process with a non-Markovian indicator function that measures success on a given episode. The episodes of τ *must* be sampled using 500 sequential states, beginning with a random initial state drawn from a continuous uniform distribution defined by the benchmark, as well as a randomized goal (this configuration is sometimes referred to as MT10-rand or MT50-rand in the literature). We refer to using the tasks from MT10 augmented with language as MT10-language, and equivalently with MT50 and MT50-language. Note that MT10 is a subset of MT50.

IV. Method

In this section, we propose a method for controlling a robot that makes use of language. We will later evaluate this method using two of the tools described above in Language-World: the task descriptions and query answering function (QAF).

Core Idea: The core idea of our method is to query the large language model (LLM) once per task to produce a fixed mapping of queries to skills (a "conditional plan"). Then, at each state, we will use a query evaluation module to evaluate each of those queries and perform a skill that corresponds to the true queries. By designing a neural architecture that interprets conditional plans in an end-to-end differentiable way, we are able to finetune the behavior of the generated plans from demonstrations, which we call plan conditioned behavioral cloning (PCBC) and describe in detail in Section IV-C.

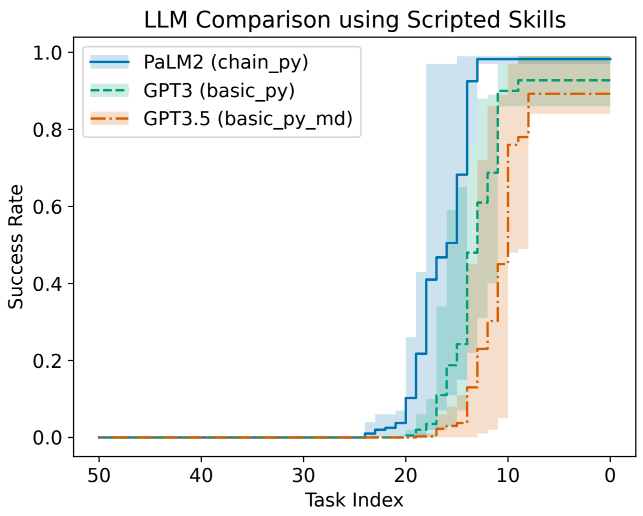

Fig. 2: This figure shows the performance of different LLM's on MT50-language using the scripted skills from MT10-language as a cumulative distribution function over tasks. Conditional plans were evaluated using the method described in Section IV-B, and this figure shows the range of performance across 4 plans per task for each LLM using the plan format that performed best with that LLM. The LLMs are able to generalize to 5-10 additional tasks outside of MT10-language, despite using only scripted skills, with PaLM 2 consistently reaching 100% for 15 total tasks.

Besides the strong experimental results we present in Section IV-D, plan conditioning has several promising conceptual aspects. By its design, plan conditioning splits the end-to-end problem into three stages, while preserving the ability for end-to-end training. The first stage, plan generation (which we describe in Section IV-A), corresponds approximately to "task planning," and can be performed before a task is attempted, allowing oversight or input from a human operator. The second stage, query evaluation, permits significant implementation flexibility. Query answering could be performed by a visual question answer (VQA) model or using value functions (as in [3]) and finetuned as part of end-to-end training. Alternatively, query answering could use any of a variety of perception methods that have been well-studied in the robotics literature. As our experiments show, constraining the generated queries to those that the query-answering module is engineered to perform can be highly effective. This allows leveraging "internet scale" models without discarding the significant progress made in robotics perception methods. The third stage, action decoding, could also be performed by a variety of methods, allowing for much higher control frequencies than the query answering module.

A. Conditional Plan Generation

In order to use a conditional plan to solve a task, we first must generate that plan. Formally, we consider a conditional plan P_τ that describes how to perform some task τ to be a set of (condition, skill) tuples (c_i, k_i), where each k_i is a natural language description of a skill, which we will embed into a continuous latent space. Depending on the benchmark c_i is either a semi-structured condition (in Language-World), or a natural language condition that can be evaluated by a visual question answering (VQA) model. Because LLMs only generate text, in order to use them to generate a conditional plan we need a *plan format* that will allow encoding and decoding conditional plans from text. We experiment with nine different plan formats, and found that different plan formats perform best for different LLMs. For example, GPT-3.5, which has been finetuned on markdown format text, performs best with `basic_py_md`.

To produce a prompt given a plan format, we first manually wrote plans for each of the tasks in MT10-language. Then, we generated a prompt for each task in MT50-language by taking manually written plans from three other tasks, and formatting them using the plan format. We chose the three other tasks by using `pick-place` (because it contains a set of skills consistently useful across many tasks), as well as the two other tasks which had task descriptions with the smallest string edit distance from the description of the task we were prompting the LLM to generate a plan for. When prompting, we never provided a plan for the task we were currently prompting for, to avoid the LLM repeating back the manually written plan. We end the prompt with the name of the task as well as the task description, which is also encoded using the plan format. We prompted each LLM four times for each combination of task and plan format.

B. LLM Experiments

In order to efficiently evaluate the performance of different combinations of LLM and plan format, we ran each generated plan using the Language-World query answering function (QAF) and scripted skills. Because the QAF only supports a finite set of (several hundred) semi-structured natural language queries, and the scripted skills consist of only 30 skills, this required us to map each condition c_i to the nearest query q_i supported by the QAF, and each natural language skill k_i to the scripted skill with the nearest description. We experimented with using both distance in an embedding space as well as using string edit distance. Because all LLMs fairly consistently matched the expected format, we found that edit distance performed sufficiently well, and used that to perform this mapping. The best results for each LLM using these plans are shown in Figure 2. In Figure 4 we compare different plan formats using PaLM 2, showing the importance of careful prompting to achieve the best results.

We call the best plan format we found `chain_py`, which uses a chain-of-thought [35] style prompt, with conditions and skills both encoded in a format that appears similar to python code. An example of a plan in this format can be seen in Figure 3.

```python
# pick-place: pick up the puck and hold it at the target location
def pick_place(robot):
    # Steps:
    #  1. Put gripper above puck
    #  2. Drop gripper around puck
    #  3. Close gripper around puck
    #  4. Move puck to goal
    # First, put the gripper roughly above puck, so that we don't
    # bump it while trying to grab it.
    if check("the robot's gripper is not above the puck"):
        robot.place("gripper above puck")
    if check("the robot's gripper is not around puck and \
            the robot's gripper is open"):
        robot.drop("gripper around puck")
    # If the gripper is near the puck and open, maybe we can grab
    # it by closing the gripper.
    if check("the robot's gripper is near puck and \
            the robot's gripper is open"):
        robot.close("gripper around puck")
    # We closed the gripper, and the puck is still near the
    # gripper, so maybe we grabbed it.
    # Try to move the puck to the goal.
    # If we didn't grab it, we'll just go back to an earlier step.
    if check("the robot's gripper is above puck and \
            the robot's gripper is closed"):
        robot.place("puck at goal")
```

Fig. 3: An example of the `pick-place` plan in the `chain_py` format. Although formatted as code, we do not evaluate this code directly. In Section IV-C we describe how to evaluate this code using PCBC, which allows finetuning the behavior of this program using end-to-end demonstrations. Our method extracts the condition in each if statement, and transform each function call into a skill description by turning the code into equivalent natural language. For example, `robot.place("gripper above puck")` becomes the skill description “place the gripper above the puck,” via a simple regex search and replace.

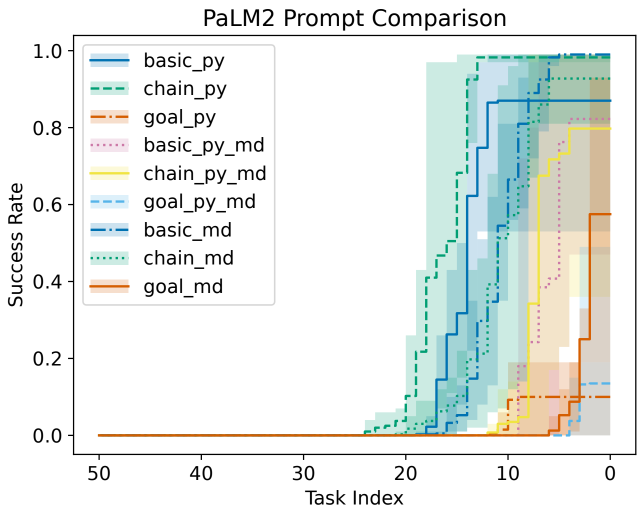

Fig. 4: This figure shows the performance of PaLM 2 using different plan formats on MT50-language using the scripted skills from MT10-language as a cumulative distribution function over tasks. Plan formats have a significant effect on performance, varying the LLM from being able to barely perform 3 tasks to being able to reliably perform 15 tasks. Shaded region is between minimum and maximum performance across 4 plans produced by the LLM for each task.

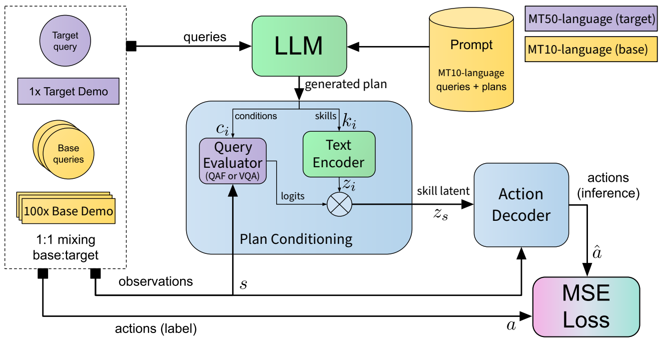

Fig. 5: This figure shows our proposed neural architecture and training setup on Language-World. The data setup for one-shot training is shown on the left. PCBC finetunes the Action Decoder to match demonstrations using the MSE Loss, and produces gradients that could be used to tune the QAF.

C. Plan Conditioning

Running a conditional plan with a set of scripted skills does produce some amount of zero-shot generalization. However, we would also like to be able to combine the advantages of conditional plans with end-to-end machine learning—without requiring scripting skills. To that end, we define an end-to-end differentiable neural architecture that interprets a conditional plan as follows. To perform a task τ using a conditional plan P_τ, first we encode each skill description k_i into a *skill latent* vector z_i. Then, to select an action on a particular state s, we evaluate each condition c_i, to form an attention vector across each skill latent z_i, and use the softmax of that attention vector to mix the skill latents into a single skill latent z_s. That skill latent z_s is then provided with the current state s to an action decoder, which produces an action for the state. The skill encoding and action decoder are trained using data from multiple tasks, as described in Section IV-E. See Figure 5 for a visual depiction of this process.

D. Conditional Plan Experiments

One of the most promising aspects of LLMs is their few-shot generalization capabilities. In this section we present experiments that demonstrate PCBC's ability to extend this few-shot generalization into the robotics domain, and compare against an architecture that produces actions conditioned on only the task description and state, which we call descriptor conditioning (DC). These experiments use the best of four plans generated with the best (plan format, LLM) combination found in Section IV-B. We run both neural architectures in three different data configurations shown in Table II.

Configuration	MT10 Demos.	MT50 Demos.	#Models	Scripted Skills
scripted skills	0	0	0	Yes
zero-shot	100 per task	0	1	No
few-shot	0	10 per task	1	No
one-shot	100 per task	1 per task	50	No

TABLE II: Data used in Figures 6 and 7. We used the same data pipeline, loss function, and optimizer for PCBC and DC. In all cases this is significantly less data than typically used by RL algorithms, which often require at least 10,000 episodes per task to achieve a non-zero success rate.

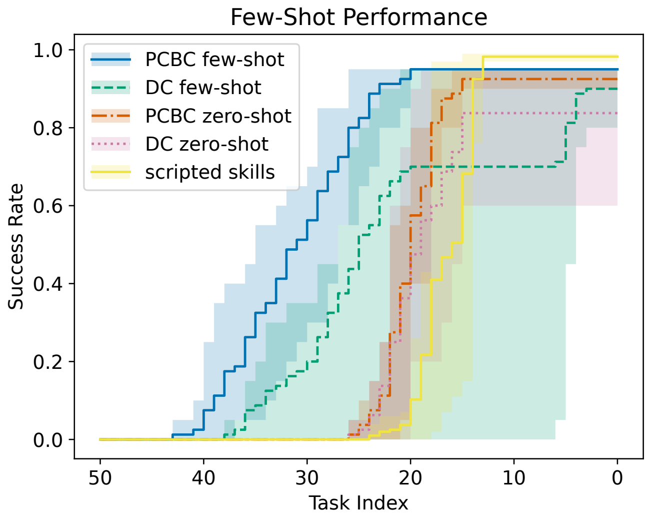

Fig. 6: This figure shows both few-shot and zero-shot performance of plan conditioned behavioral cloning (PCBC) as well as descriptor conditioning (DC) and scripted skills on MT50-language. Runs marked zero-shot were pre-trained from 100 demonstrations of MT10-language and were only provided a task description for MT50-language. Runs marked few-shot received 10 demonstrations for each task in MT50-language, as well as a task description. In both cases, one "universal" policy is learned for all tasks. End-to-end training improves over scripted skills, and plan conditioning (PCBC) maintains a higher consistent level of performance across many tasks than descriptor conditioning (DC). Shaded region is between min and max performance across 4 seeds.

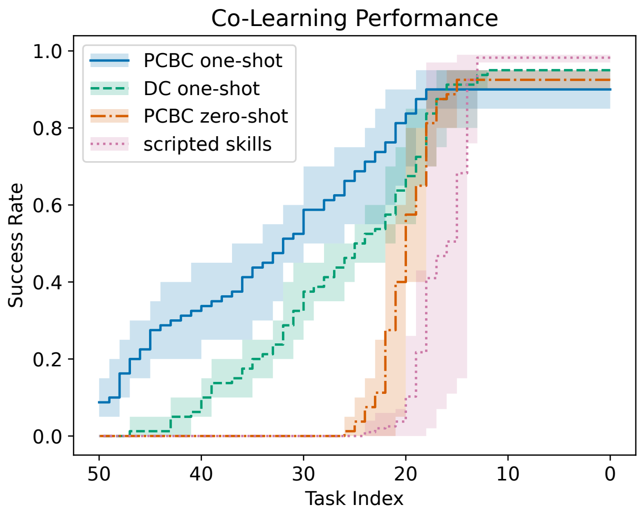

Fig. 7: This figure shows how adding a single demonstration to the zero-shot setting described in Figure 6 results in a significant increase in performance across several tasks, and non-zero performance on every task, despite each task having randomized goal locations and initial states. The single demonstration of the MT50-language task is combined with 100 demonstrations of each MT10-language task using the co-learning method described in Section IV-E to train a single model for each task. Shaded region is between minimum and maximum performance across 4 seeds.

E. *Cross-Task Co-Learning via 1:1 Data Mixing*

In zero-shot and few-shot training, we train a single "universal" policy to perform all tasks. We do this by training on minibatches with an equal number of (task, state, action) tuples from each task. In the one-shot data configuration, we instead seek to train a separate model for each *target task* from a single demonstration of that task by leveraging demonstrations of other *base tasks*. We achieve this by using minibatches which contain a mix of tuples sampled in equal number from all base tasks and an equal number of tuples sampled from the single target tasks demonstration, as shown in Figure 8. This follows prior work which found such 1:1 (one-to-one) data mixing to be effective in deep Q learning methods [12], [16].

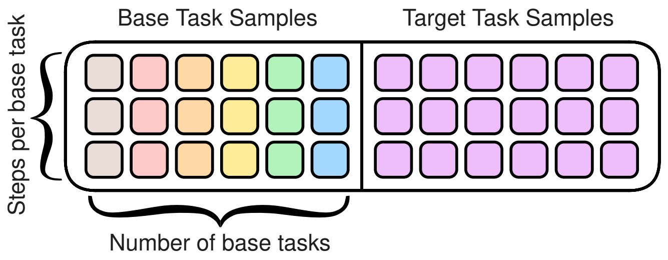

Fig. 8: A co-learning minibatch used in one-shot training. Each cell contains a (task, state, action) tuple which is used with the end-to-end BC loss to optimize the policy. Because there are significantly more target task tuples than tuples for any particular base task, the model will primarily be optimized for the target task while being regularized by the base task demonstrations. This regularization allows the trained policy to be robust to randomizations in the initial and goal state of a task, despite being trained on only a single demonstration with only a single initial state and goal state, as shown in Figure 7. Co-learning is able to achieve this generalization without making strong assumptions about the structure of the observation or action space.

F. *Differentiability and Optimization*

PCBC essentially splits action selection into three steps: plan generation, query evaluation, and action decoding. In our experiments using Language-World, query evaluation is performed with the query answering function, so only the action decoder is finetuned with gradient descent. In a real-world application of PCBC using a visual question answer (VQA) model in place of the QAF, both of the VQA model and action decoder could be tuned with gradient descent. Our experiments also perform both query evaluation and action decoding at each timestep. Although this already uses significantly less computational resources than evaluating a language model at each timestep, it may be possible to use fewer still computational resources by performing query evaluation at a lower frequency than every timestep. Because the action decoder is trained as part of the end-to-end objective, it is able to compensate for minor inconsistencies in the transition between skills, such as those a lag in evaluating queries would introduce.

In this work we relied on the implicit regularization of small batch sizes (less than 200 total timesteps per minibatch) to avoid overfitting to our small dataset. In future work, it would be worthwhile to combine our method with regularization or contrastive learning techniques that may allow improving generalization further while learning with larger batch sizes.

V. Limitations

a) ***Reinforcement Learning****:* In this work we chose to focus on using imitation learning, because PCBC required only a very small number of expert demonstrations to reach high performance. However, significant prior research exists in using reinforcement learning (RL) for robotic control, and Meta-World (which Language-World extends), is designed as a (Meta/Multi-Task) RL benchmark. In future work, it would be worthwhile to experiment with using plan conditioning and RL, and Offline RL in particular.

b) ***Plan Quality****:* In this work we use plans generated by large language models (LLMs). Although we experimented with a variety of plan formats, and ran each LLM multiple times for each (task, plan format) combination, many of the generated plans were still low quality. We expect that further improvements in prompting methods, which is an area of active research, may be able to improve our one-shot results further. Alternatively, writing more plans by hand, either to use directly or to fine-tune an LLM, may be effective.

c) ***Plan Complexity****:* In this work we explored conditioning on the steps of a plan. This is similar to evaluating a single switch statement in an end-to-end differentiable way. In future work, we intend to explore using end-to-end differentiable models of more complex computational constructs.

VI. Acknowledgements

We would like to thank people at Google who gave feedback on earlier versions of this work. In particular, Yevgen Chebotar, Karol Hausman, Chelsea Finn, Sergey Levine, Fei Xia, and Alex Irpan.

VII. Conclusion

In this work, we introduced a new benchmark, Language-World, which uses the same task as the popular Meta-World benchmark, but extends it to allow it to be easily used in experiments with large language models. We also introduced a method, plan conditioned behavioral cloning (PCBC), which serves as a strong baseline imitation learning method for Language-World. By leveraging Cross-Task Co-Learning, PCBC is able to achieve promising performance from a single demonstration per task, and extremely strong performance at 100 demonstrations. In all cases, PCBC uses significantly less data than typically used by RL algorithms on these same tasks, which often require at least 10,000 episodes to achieve a non-zero success rate on a single task.

REFERENCES

[1] Google ai palm 2. `https://ai.google/discover/palm2/`. Accessed: 2023-09-14.

[2] Gpt-3.5. `https://platform.openai.com/docs/models/gpt-3-5`. Accessed: 2023-09-14.

[3] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can and not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

[4] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022.

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam M. Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Benton C. Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier García, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Díaz, Orhan Firat, Michele Catasta, Jason Wei, Kathleen S. Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *ArXiv*, abs/2204.02311, 2022.

[7] Felipe Codevilla, Matthias Müller, Alexey Dosovitskiy, Antonio M. López, and Vladlen Koltun. End-to-end driving via conditional imitation learning. *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–9, 2017.

[8] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, and Joelle Pineau. An introduction to deep reinforcement learning. *arXiv preprint arXiv:1811.12560*, 2018.

[9] Vijaykumar Gullapalli, Judy A Franklin, and Hamid Benbrahim. Acquiring robot skills via reinforcement learning. *IEEE Control Systems Magazine*, 14(1):13–24, 1994.

[10] Leonard Hasenclever, Fabio Pardo, Raia Hadsell, Nicolas Heess, and Josh Merel. CoMic: Complementary task learning & mimicry for reusable skills. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 4105–4115. PMLR, 13–18 Jul 2020.

[11] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. In *International Conference on Learning Representations*, 2018.

[12] Ryan Julian, Benjamin Swanson, Gaurav S Sukhatme, Sergey Levine, Chelsea Finn, and Karol Hausman. Never stop learning: The effectiveness of fine-tuning in robotic reinforcement learning. *arXiv e-prints*, pages arXiv–2004, 2020.

[13] Ryan C Julian, Eric Heiden, Zhanpeng He, Hejia Zhang, Stefan Schaal, Joseph Lim, Gaurav S Sukhatme, and Karol Hausman. Scaling simulation-to-real transfer by learning composable robot skills. In *International Symposium on Experimental Robotics*. Springer, 2018.

[14] Jens Kober, J Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[15] Oliver Kroemer, Christian Daniel, Gerhard Neumann, Herke Van Hoof, and Jan Peters. Towards learning hierarchical skills for multi-phase manipulation tasks. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1503–1510. IEEE, 2015.

[16] Alex X. Lee, Coline Devin, Jost Tobias Springenberg, Yuxiang Zhou, Thomas Lampe, Abbas Abdolmaleki, and Konstantinos Bousmalis. How to spend your robot time: Bridging kickstarting and offline reinforcement learning for vision-based robotic manipulation, 2022.

[17] Tianyu Li, Nathan Lambert, Roberto Calandra, Franziska Meier, and Akshara Rai. Learning generalizable locomotion skills with hierarchical reinforcement learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 413–419. IEEE, 2020.

[18] Jacky Liang, Wenlong Huang, F. Xia, Peng Xu, Karol Hausman, Brian Ichter, Peter R. Florence, and Andy Zeng. Code as policies: Language model programs for embodied control. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9493–9500, 2022.

[19] Long-Ji Lin. Reinforcement learning for robots using neural networks, 1992.

[20] Sridhar Mahadevan and Jonathan Connell. Automatic programming of behavior-based robots using reinforcement learning. *Artificial intelligence*, 55(2-3):311–365, 1992.

[21] Josh Merel, Saran Tunyasuvunakool, Arun Ahuja, Yuval Tassa, Leonard Hasenclever, Vu Pham, Tom Erez, Greg Wayne, and Nicolas Heess. Catch & carry: reusable neural controllers for vision-guided whole-body tasks. *ACM Transactions on Graphics (TOG)*, 39(4):39–1, 2020.

[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[23] OpenAI. Gpt-4 technical report, 2023.

[24] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.

[25] Peter Pastor, Mrinal Kalakrishnan, Ludovic Righetti, and Stefan Schaal. Towards associative skill memories. In *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pages 309–315, November 2012.

[26] Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. Mcp: Learning composable hierarchical control with multiplicative compositional policies. *Advances in Neural Information Processing Systems*, 32, 2019.

[27] Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, Yi Ren Fung, Yusheng Su, Huadong Wang, Cheng Qian, Runchu Tian, Kunlun Zhu, Shi Liang, Xingyu Shen, Bokai Xu, Zhen Zhang, Yining Ye, Bo Li, Ziwei Tang, Jing Yi, Yu Zhu, Zhenning Dai, Lan Yan, Xin Cong, Ya-Ting Lu, Weilin Zhao, Yuxiang Huang, Jun-Han Yan, Xu Han, Xian Sun, Dahai Li, Jason Phang, Cheng Yang, Tongshuang Wu, Heng Ji, Zhiyuan Liu, and Maosong Sun. Tool learning with foundation models. *ArXiv*, abs/2304.08354, 2023.

[28] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2023.

[29] E. Rueckert, J. Mundo, A. Paraschos, J. Peters, and G. Neumann. Extracting low-dimensional control variables for movement primitives. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1511–1518, May 2015.

[30] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023.

[31] William D Smart and L Pack Kaelbling. Effective reinforcement learning for mobile robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 4, pages 3404–3410. IEEE, 2002.

[32] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[33] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Huai hsin Chi, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *ArXiv*, abs/2203.11171, 2022.

[34] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. Describe, explain, plan and select: Interactive planning with large language models enables open-world multi-task agents. *ArXiv*, abs/2302.01560, 2023.

[35] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *ArXiv*, abs/2201.11903, 2022.

[36] Markus Wulfmeier, Dushyant Rao, Roland Hafner, Thomas Lampe, Abbas Abdolmaleki, Tim Hertweck, Michael Neunert, Dhruva Tirumala, Noah Siegel, Nicolas Heess, et al. Data-efficient hindsight off-policy option learning. In *International Conference on Machine Learning*, pages 11340–11350. PMLR, 2021.

[37] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. Multi-task reinforcement learning with soft modularization. *Advances in Neural Information Processing Systems*, 33:4767–4777, 2020.

[38] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on Robot Learning (CoRL)*, 2019.

[39] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Ken Goldberg, and P. Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *IEEE International Conference on Robotics and Automation*, 2017.

[40] Denny Zhou, Nathanael Scharli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Olivier Bousquet, Quoc Le, and Ed Huai hsin Chi. Least-to-most prompting enables complex reasoning in large language models. *ArXiv*, abs/2205.10625, 2022.

[41] Xuefeng Zhou, Hongmin Wu, Juan Rojas, Zhihao Xu, and Shuai Li. Incremental learning robot task representation and identification. In *Nonparametric Bayesian Learning for Collaborative Robot Multimodal Introspection*, pages 29–49. Springer, 2020.