CALVIN: A Benchmark for Language-Conditioned Policy Learning for Long-Horizon Robot Manipulation Tasks

Oier Mees*[1], Lukas Hermann*[1], Erick Rosete-Beas[1], Wolfram Burgard[2]
http://calvin.cs.uni-freiburg.de

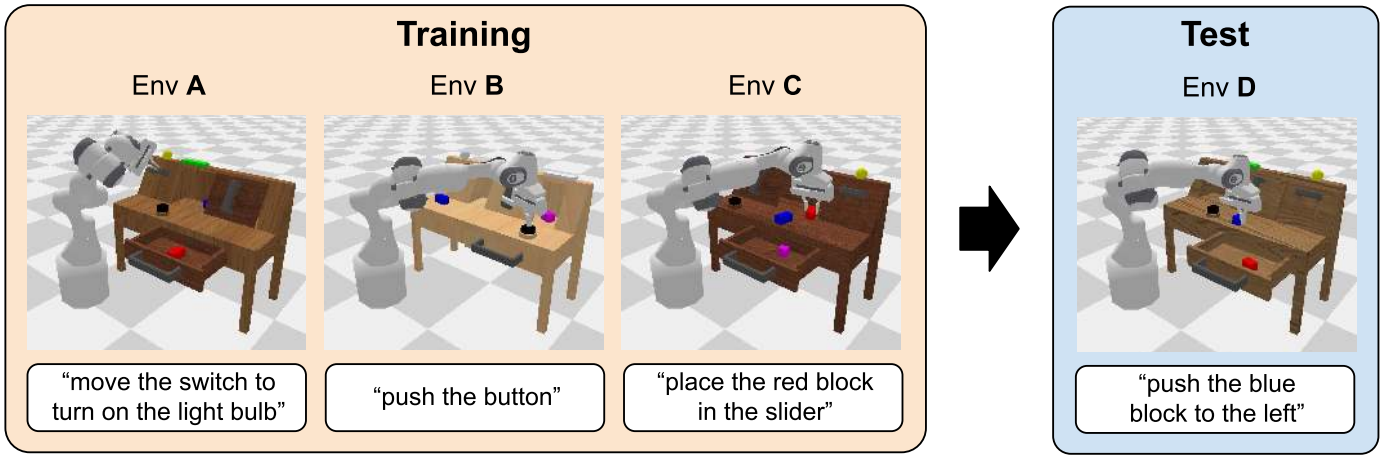

Fig. 1: CALVIN is a benchmark to learn many long-horizon language-conditioned tasks over a range of four manipulation environments, designed to be diverse yet carry shared structure, from multimodal onboard sensor observations. In the most difficult evaluation, the methods must generalize to unseen entities by training on a large interaction corpora covering three environments and testing on an unseen scene.

***Abstract*—General-purpose robots coexisting with humans in their environment must learn to relate human language to their perceptions and actions to be useful in a range of daily tasks. Moreover, they need to acquire a diverse repertoire of general-purpose skills that allow composing long-horizon tasks by following unconstrained language instructions. In this paper, we present CALVIN (Composing Actions from Language and Vision), an open-source simulated benchmark to learn long-horizon language-conditioned tasks. Our aim is to make it possible to develop agents that can solve many robotic manipulation tasks over a long horizon, from onboard sensors, and specified only via human language. CALVIN tasks are more complex in terms of sequence length, action space, and language than existing vision-and-language task datasets and supports flexible specification of sensor suites. We evaluate the agents in zero-shot to novel language instructions and to novel environments. We show that a baseline model based on multi-context imitation learning performs poorly on CALVIN, suggesting that there is significant room for developing innovative agents that learn to relate human language to their world models with this benchmark.**

Manuscript received: February, 23, 2022; Accepted May, 22, 2022.
This paper was recommended for publication by Associate Editor S. Chernova and Editor D. Kulic upon evaluation of the reviewers' comments.
*Equal contribution.[1]University of Freiburg, Germany.[2]University of Technology Nuremberg, Germany. meeso@informatik.uni-freiburg.de
Digital Object Identifier (DOI): see top of this page.

***Index Terms*—Data Sets for Robot Learning, Machine Learning for Robot Control, Imitation Learning, Natural Dialog for HRI**

I. Introduction

A LONG-STANDING goal for robotics and embodied agents is to build systems that can perform tasks specified in natural language. Concepts expressed in natural language provide humans with an intuitive way to represent, summarize, and abstract diverse knowledge skills. By means of abstraction, concepts such as "open the drawer and push the middle object into the drawer" can be extended to a potentially infinite set of new and unseen entities. Additionally, humans leverage concepts to describe complex tasks as sequences of natural language instructions. This stands in contrast to current robots, which typically lack this generalization ability and learn individual tasks one at a time. Moreover, multi-task learning approaches traditionally assume that tasks are specified to the agent at test time via mechanisms such as goal images [1] and one-hot skill selectors [2], [3] that are not practical for non-expert users to instruct robots in everyday real-world settings. As robots become ubiquitous across human-centered environments the need for intuitive task specification grows: how can we scale robot learning systems to autonomously acquire general-purpose knowledge that allows them to compose long-horizon tasks by following unconstrained language instructions?

To address this problem we present CALVIN, a new open-source simulated benchmark that links human language to robot motor skills, behaviors, and objects in interactive visual environments. In this setting, a single agent must solve complex manipulation tasks by understanding a series of unconstrained language expressions in a row, e.g., "open the drawer ... pick up the blue block ... push the block into the drawer ... open the sliding door". Furthermore, to evaluate the agents' ability for long-horizon planning, agents in this scenario are expected to be able to perform any combination of subtasks in any order. CALVIN has been developed from the ground up to support training, prototyping, and validation of language-conditioned continuous control policies over a range of four indoor manipulation environments, visualized in Figure 1. CALVIN includes $\sim$24 hours teleoperated unstructured *play* data together with 20K language directives. Unscripted playful interactions have the advantage of being task-agnostic, diverse, and relatively cheap to obtain [1], [4]. The simulation platform supports a range of sensors commonly utilized for visuomotor control: RGB-D images from both a static and a gripper camera, proprioceptive information, and vision-based tactile sensing [5]. We believe that this flexible sensor suite will allow researchers to develop improved multimodal agents that can solve many tasks in real-world settings. This is the first public benchmark of instruction following, to our knowledge, that combines: natural language conditioning, multimodal high-dimensional inputs, 7-DOF continuous control, and long-horizon robotic object manipulation. We provide an evaluation protocol with evaluation modes of varying difficulty by choosing different combinations of sensor suites and amounts of training environments. This effort joins the recent efforts to standardize robotics research for better benchmarks and more reproducible results. To open the door for future development of agents that can generalize abstract concepts to unseen entities the same way humans do, we include a challenging zero-shot evaluation by training on large play corpora covering three environments and testing on an unseen scene. The language instructions used for testing are not included in the training set and represent novel ways of describing the manipulation tasks seen during training.

To establish baseline performance levels, we evaluate the multi-context imitation learning (MCIL) approach that uses relabeled imitation learning to distill many reusable behaviors into a goal-directed policy [6]. This model is not effective on the complex long horizon robot manipulation tasks in CALVIN. While it achieves up to 53.9% success rate in short horizon tasks, it performs poorly in the long-horizon setting. We note that there is no constraint to use imitation learning approaches to solve CALVIN tasks, as approaches that use reinforcement learning to learn language-conditioned policies can also be applied [7].

In summary, CALVIN facilitates learning models that translate from language to sequences of motor skills in a realistic simulation environment. This benchmark captures many challenges present in real-world settings for relating human language to robot actions and perception for accomplishing long-horizon manipulation tasks. Models that can overcome these challenges will begin to close the gap towards scalable, general-purpose, language-driven robotics.

II. Related Work

Natural language processing has recently received much attention in the field of robotics [8], following the advances made towards learning groundings between vision and language [9], [10]. Recent successes in human-robot interaction include an interactive fetching system to localize objects mentioned in referring expressions [11]–[15] or grounding not only objects, but also spatial relations to follow language expressions characterizing pick-and-place commands [16]–[18]. By contrast, CALVIN tasks require grounding language to a wide variety of general-purpose robot skills. Prior work on mapping language and vision to actions has been studied mostly in restricted environments [19], [20] and simplified actuators with discrete motion primitives [21]–[23]. A growing body of work also looks at learning language-conditioned policies for continuous visuomotor-control in 3D environments via imitation learning [6], [24], [25] or reinforcement learning [7], [26], [27]. These approaches typically require offline data sources of robotic interaction, such as teleoperation or autonomous exploration data, together with post-hoc crowd-sourced language labels. However, the lack of standardized benchmarks and algorithm implementations, makes it difficult to compare approaches and to facilitate future research.

The most closely related benchmark to ours is ALFRED [22], which contains language instructions for combined navigation and manipulation tasks with seven predefined action primitives. In CALVIN, rather than classifying predefined actions, the agent must learn to acquire a diverse repertoire of general-purpose skills that allows composing long-horizon tasks by following unconstrained language instructions in closed loop control. Our tabletop environments are inspired by the one shown in Lynch *et al.* [6] in order to have a fair comparison to their MCIL approach, which we implement to establish baseline performance levels. We note that although considered a state-of-the-art approach, no public implementation of MCIL is available. In contrast to their work, CALVIN contains more subtasks (34 vs 18), longer long-horizon evaluation sequences (5 vs 4), provides a range of sensors commonly utilized for visuomotor control and allows testing zero-shot generalization by leveraging a range of four manipulation environments and unseen language instructions. Finally, CALVIN goes beyond the original MCIL setup by adding a challenging visual grounding problem, where similar language instructions for differently colored blocks are given and the agent needs to identify which block is meant.

III. CALVIN

The aim of the CALVIN benchmark is to evaluate the learning of long-horizon language-conditioned continuous control policies. In this setting, a single agent must solve complex manipulation tasks by understanding a series of unconstrained language expressions in a row, e.g., "open the drawer... pick up the blue block... now push the block into the drawer... now open the sliding door". We note that in the benchmark we

Observation Space	
RGB static camera	$200 \times 200 \times 3$
Depth static camera	200×200
RGB gripper camera	$84 \times 84 \times 3$
Depth gripper camera	84×84
Tactile image	$120 \times 160 \times 2$
Proprioceptive state	EE position (3) EE orientation (3) Gripper width (1) Joint positions (7) Gripper action (1)
Action Space	
Absolute cartesian pose (w.r.t. world frame)	EE position (3) EE orientation (3) Gripper action (1)
Relative cartesian displacement (w.r.t. gripper frame)	EE position (3) EE orientation (3) Gripper action (1)
Joint action	Joint positions (7) Gripper action (1)

Fig. 2: Observation and action spaces supported by CALVIN.

only allow feasible sequences that can be achieved from a predefined initial environment state. The CALVIN benchmark consists of three key components, which are:

1) CALVIN Environment
2) CALVIN Dataset
3) CALVIN Challenge

A. The CALVIN Environment

CALVIN features four different, yet structurally related environments (A, B, C, D) so that it can be used for general playing as well as evaluating specific tasks. The environments contain a 7-DOF Franka Emika Panda robot arm with a parallel gripper and a desk with a sliding door and a drawer that can be opened and closed. On the desk, there is a button that toggles a green light and a switch to control a light bulb. Besides, there are three different colored and shaped rectangular blocks. To better evaluate the generalization capabilities of the learned language groundings, all environments have different textures and all static elements such as the sliding door, the drawer, the light button, and switch are positioned differently. The position of the desk, robot, and the static camera is the same in all environments. Due to the general difficulty of language-conditioned multi-task closed-loop control, we reduced the complexity of the objects to unicolored primitive shapes. If future advances in this field require new challenges we will reflect this by extending CALVIN to environments with more realistic and diverse objects. Physics are simulated using the PyBullet physics engine [28], which supports fast GPU rendering for large-scale parallel data collection.

1) Observation and Action Space: Unlike prior work which relies on RGB images from an egocentric camera to perceive its surroundings [1], [6], CALVIN offers a range of sensors that can be used to develop and prototype agents that learn task-agnostic control in the real world. Concretely, the agent

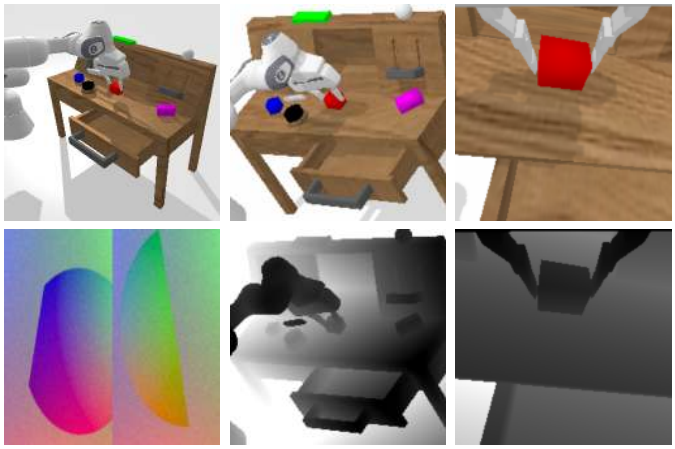

Fig. 3: CALVIN supports a range of sensors commonly utilized for visuomotor control: RGB-D images from both a static and a gripper camera, proprioceptive information, and vision-based tactile sensing (bottom-left).

perceives its surroundings from RGB-D images from both a fixed and a gripper camera. It additionally has access to a vision-based tactile sensor [5] and to continuous internal proprioceptive sensors. A visualization of the supported sensor modalities is shown in Figure 3. The agent must perform closed-loop continuous control to follow unconstrained language instructions characterizing complex robot manipulation tasks, sending continuous actions to the robot at 30hz. In order to give researchers and practitioners the freedom to experiment with different action spaces, CALVIN supports absolute and relative cartesian actions, as well as actions in joint space. We encourage the community to study flexible combinations of observation and action spaces since the tasks require a varying degree of precise control vs. coarse locomotion. While the static camera and absolute cartesian actions are the natural choices for tasks that call for a complete traversal of the environment from one side to another, the gripper camera and relative actions (w.r.t to the gripper frame) allow more fine-grained control for tasks like stacking or grasping. Tactile information can become important when the task requires the robot to maintain a stable grasp on the handle while moving the sliding door to the side. See Fig. 2 for a description of the observation and action dimensionalities.

2) Tasks: We define 34 specific tasks (see Fig. 4) that can be achieved in each one of the environments The environment has the functionality to automatically detect which one of the tasks has been completed in a sequence of steps, which can serve as a sparse reward for reinforcement learning agents. The criterion for task completion is defined in terms of a change in the environment state between the initial and final step of a sequence. This also enables the automatic task detection in any variable-length sequence of offline data, since the environment can be reset to the state of each one of the recorded frames.

B. The CALVIN Dataset

1) Unstructured Demonstrations: Learning generally requires exposure to diverse training data. To effectively cover state space, we collect twenty-four hours of teleoperated "play"

Task	Natural language instructions
rotate red block right	"rotate the red block 90 degrees to the right" "turn the red block right"
push blue block left	"go slide the blue block to the left" "push left the blue block"
move slider left	"grasp the door handle, then slide the door to the left" "slide the door to the left"
open drawer	"grasp the handle of the drawer and open it" "go open the drawer"
lift red block	"lift the red block from the table" "pick up the red block"
pick pink block from drawer	"pick up the pink block lying in the drawer"
place in slider	"put the grasped object in the slider"
stack blocks	"stack blocks on top of each other"
unstack blocks	"collapse the stacked blocks" "go to the tower of blocks and take off the top one"
turn on light bulb	"toggle the light switch to turn on the light bulb"
turn off green light	"push the button to turn off the green light"

Fig. 4: Example crowd-sourced natural language instructions to specify manipulation tasks in CALVIN.

data in four environments with a HTC Vive VR headset, spending an approximately equal time of six hours in each environment. This corresponds to ∼2.4M interaction steps and ∼40M short-horizon windows for relabeled goal conditioned imitation learning [29], [30], each spanning 1-2 seconds. In this setting, an operator is not constrained to a set of predefined tasks, but rather engages in behavior that satisfies their own curiosity or some other intrinsic motivation. Unscripted playful interactions have the advantage of being task-agnostic, diverse, and relatively cheap to obtain [1], [4]. We asked three people to collect data, and these users were untrained and given no information about the downstream tasks. The only guideline we gave data collectors was to "explore the environment without dropping objects from the table". This includes picking up and placing objects, opening, and closing drawers, sliding doors, pushing buttons, operating switches and undirected actions. This style of data is very different from commonly used task-specific data, which only consists of expert trajectories. Playful interaction data by design is free-form, so there are no categories associated with the data. This kind of unstructured data is useful because it contains exploratory and sub-optimal behaviors that are critical to learning generalizable and robust representations, e.g., enabling retrying behavior. While expert demonstrations often only show one of the many possible ways to solve a task, play data is richer in the sense that it covers the multimodal space of possible solutions. However, as opposed to expert demonstrations, in play data some task instances naturally occur less frequently than others, especially those that have the completion of another task as a prerequisite.

2) Language Instructions: Approaches that learn language-conditioned continuous control policies typically require post-hoc crowd-sourced natural language labels aligned with its corresponding robot interaction data [6], [7]. Instead of relying entirely on crowd-sourced annotations, we collect over 400 crowd-sourced natural language instructions corresponding to over 34 tasks and label episodes procedurally using the recorded environment state of the CALVIN dataset. We note that using this labeling scheme, only sequences that display meaningful skills are labeled with language annotations. We visualize example language annotations in Fig. 4. In order to simulate a real-world scenario where it might not be possible to pair all the collected robot experience with crowd-sourced language annotations, we annotate only 1% of the recorded robot interaction data with language instructions. Besides language instructions, we provide precomputed language embeddings extracted from MiniLM [31]. MiniLM distills a large Transformer based language model and is trained on generic language corpora (e.g., Wikipedia). It has a vocabulary size of 30,522 words and maps a sentence of any length into a vector of size 384. We note that there exist many choices for encoding raw text in a semantic pre-trained vector space and encourage the community to experiment with different choices to solve for CALVIN tasks.

C. The CALVIN Challenge

CALVIN combines the challenging settings of open-ended robotic manipulation with open-ended human language conditioning. For example, a robot that is instructed to "place the blue block inside the drawer" must be able to relate language to its world model. Concretely, it needs to learn to identify how a blue block and a drawer look like in its multimodal perceptual observations[1], and then it needs to reason over the best sequence of actions to "place inside the drawer". Ideally, a general-purpose robot should be able to perform any combination of tasks instructed with natural language in any order. Thus, to accelerate progress in language-driven robotics, we present a set of evaluation protocols of varying difficulty by choosing different combinations of sensor suites and amounts of training environments.

1) Training and Test Environments: CALVIN offers three combinations of training and test environments with varying difficulty:

Single Environment: Training in a single environment and evaluating the policy in the same environment. This corresponds to the setting of Lynch *et al.* [6].

Multi Environment: Training in all four environments and evaluating the policy in one of them. This poses an additional challenge since the policy has to generalize to multiple textures

[1]Simulator states consisting of object positions and orientations are also provided, but not used to better capture challenges of real-world settings.

Long-horizon language instructions
"turn on the led" → "open drawer" → "push the blue blue block → "pick up the blue block " → "place in slider"
"move slider left" → "lift red block from slider" → "stack blocks" → "toggle light" → " collapse stacked blocks"
"open drawer" → "push block in drawer" → "pick object from drawer" → "stack blocks" → "close drawer"

Fig. 5: Example long-horizon language tasks sequences evaluated in CALVIN. We show the abbreviated subtask names instead of the full language annotations due to space constraint.

and different locations of the sliding door, button, and switch. On the other hand, the agents can benefit from increased data.

Zero-Shot Multi Environment: To open the door for future development of agents that can generalize abstract concepts to unseen entities the same way humans do, we include a challenging zero-shot evaluation by training in three environments and evaluating the policy in the fourth unseen one. This is the hardest combination since the policy has never seen the test environment during training. However, all elements of the scene were present in different locations in the training environments. While highly challenging, we believe it aligns well with test-time expectations for service robots to be useful in a range of daily tasks in everyday environments. Concretely, in CALVIN agents need to generalize to a room where the environment has different textures and all static elements such as the sliding door, the drawer and the light turning button and switch are positioned differently. Thus, a language-conditioned policy should ideally be able to open a sliding door even if it is differently positioned or looks visually a bit different.

2) Evaluation Metrics: All three environment combinations are evaluated with the following metrics:

Multi-Task Language Control (MTLC): The simplest evaluation aims to verify how well the learned multi-task language-conditioned policy generalizes to 34 manipulation tasks, which we visualize in Fig. 6. The evaluation begins by resetting the simulator to the first state of a valid unseen demonstration, to ensure that the commanded instruction is valid. For each manipulation task 10 rollouts are performed with their corresponding different starting states. The language instructions used for testing are not included in the training set and represent novel ways of describing the manipulation tasks seen during training.

Long-Horizon MTLC (LH-MTLC): This evaluation aims to verify how well the learned multi-task language-conditioned policy can accomplish several language instructions in a row. This setting is very challenging as it requires agents to be able to transition between different subgoals. We treat the 34 tasks of the previous evaluation as subgoals and compute valid sequences consisting of five sequential tasks. We only allow feasible sequences that can be achieved from a predefined initial environment state. We filter the evaluation sequences for cycles, redundancies and similarities to arrive at 1000 unique instruction chains. Examples for excluded sequences are "close the drawer"..."place in drawer" (unfeasible), "move slider right"..."move slider left"..."move slider right" (cyclic) or "push blue block left"..."push red block left"(similar). We reset the robot to a neutral position after every sequence to avoid biasing the policies through the robot's initial pose. We note that this neutral initialization breaks correlation between initial state and task, forcing the agent to rely entirely on language to infer and solve the task. We include different initial scene configurations in the evaluation to better evaluate generalization capabilities. We visualize the evaluated subtask distribution in Figure 7. For each subtask we condition the policy on the current language instruction and transition to the next subgoal only if the agent successfully completes the current task according to the environments state indicator.

Task	Condition
Rotate red/blue/pink block right	The object has to be rotated clockwise more than 60° around the z-axis while not being rotated more than 30° around the x or y-axis.
Rotate red/blue/pink block left	The object has to be rotated counterclockwise more than 60° around z while not being rotated more than 30° around the x or y-axis.
Push red/blue/pink block right	The object has to move more than 10 cm to the right while having surface contact in both frames.
Push red/blue/pink block left	The object has to move more than 10 cm to the left while having surface contact in both frames.
Move slider left/right	The sliding door has to be pushed at least 12 cm to the left/right.
Open/close drawer	The drawer has to be pushed in/pulled out at least 10 cm.
Lift red/blue/pink block table	The object has to be grasped from the table surface and lifted at least 5 cm high. In the first frame the gripper may not touch the object.
Lift red/blue/pink block slider	The object has to be grasped from the sliding cabinet's surface and lifted at least 3 cm. In the first frame the gripper may not touch the object.
Lift red/blue/pink block drawer	The object has to be grasped from the drawer's surface and lifted at least 5 cm high. In the first frame the gripper may not touch the object.
Place in slider/drawer	The object has to be placed in the sliding cabinet/drawer. It must be lifted by the gripper in the first frame.
Push into drawer	The object has to be pushed into the drawer. It has to touch the table surface in the first frame.
Stack blocks	A block has to be placed on top of another block. It may not be in contact with the gripper in the final frame.
Unstack blocks	A block has to be removed from the top of another block. It may not be in contact with the gripper in the first frame.
Turn on/off light bulb	The switch has to be pushed up/down to turn on/off the yellow light bulb.
Turn on/off LED	The button has to be pressed to turn on/turn off the green LED light.

Fig. 6: List of all 34 tasks with their respective success criteria.

3) Sensor Combinations: The aim of CALVIN is to develop innovative agents that learn to relate human language from onboard sensors by capturing many challenges present in real-world settings. Most autonomous robots operating in complex

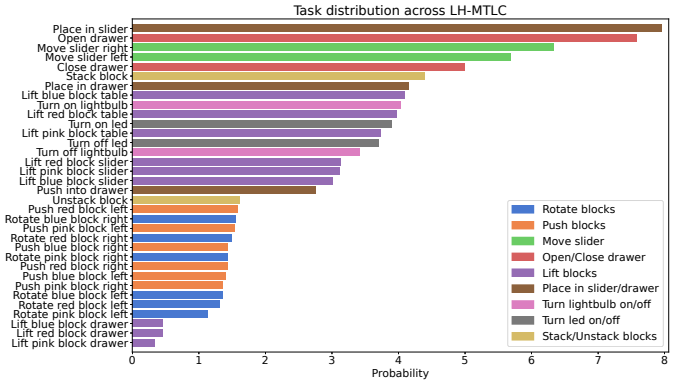

Fig. 7: Visualization of the subtask distribution across the 1000 instruction chains used for the Long Horizon MTLC evaluation. We show the percentage in which each subtask appears in the distribution.

environments are equipped with different sensors to perceive their surroundings. To foster development and experimentation of language-conditioned policies that perform manipulation tasks in the real-world, CALVIN supports a range of sensors commonly utilized for visuomotor control: RGB-D images from both a static and a gripper camera, proprioceptive information, and vision-based tactile sensing [5]. We therefore evaluate baseline agents for different sensors combinations.

IV. Baseline Models

An agent trained for CALVIN needs to jointly reason over perceptual and language input and produce a sequence of low-level motor commands to interact with the environment.

A. Multicontext Imitation Learning

We model the interactive agent with a general-purpose goal-reaching policy based on multi-context imitation learning (MCIL) from play data [6]. To learn from unstructured "play" we assume access to an unsegmented teleoperated play dataset $\mathcal{D}$ of semantically meaningful behaviors provided by users, without a set of predefined tasks in mind. To learn control, this long temporal state-action stream $\mathcal{D} = \{(x_t, a_t)\}_{t=0}^{\infty}$ is relabeled [30], treating each visited state in the dataset as a "reached goal state", with the preceding states and actions treated as optimal behavior for reaching that goal. Relabeling yields a dataset of $D_{\text{play}} = \{(\tau, x_g)_i\}_{i=0}^{D_{\text{play}}}$ where each goal state x_g has a trajectory demonstration $\tau = \{(x_0, a_0), \ldots\}$ solving for the goal. These short horizon goal image conditioned demonstrations can be fed to a simple maximum likelihood goal conditioned imitation objective:

$$\mathcal{L}_{LfP} = \mathbb{E}_{(\tau, x_g) \sim D_{\text{play}}} \left[\sum_{t=0}^{|\tau|} \log \pi_\theta(a_t \mid x_t, x_g) \right] \tag{1}$$

to learn a goal-reaching policy $\pi_\theta\left(a_t \mid x_t, x_g\right)$. Multi-context imitation learning addresses the inherent multi-modality in free-form imitation datasets by auto-encoding contextual demonstrations through a latent "plan" space with an sequence-to-sequence conditional variational auto-encoder (seq2seq CVAE). The decoder is a policy trained to reconstruct input actions, conditioned on state x_t, goal x_g, and an inferred plan z for how to get from x_t to x_g. At test time, it takes a goal as input, and infers and follows plan z in closed-loop.

However, when learning language-conditioned policies $\pi_\theta\left(a_t \mid x_t, l\right)$ it is not possible to relabel any visited state x to a natural language goal as the goal space is no longer equivalent to the observation space. Lynch *et al.* [6] showed that pairing a small number of random windows with language after-the-fact instructions enables learning a single language-conditioned visuomotor policy that can perform a wide variety of robotic manipulation tasks. The key insight here is that solving a single imitation learning policy for either goal image or language goals, allows for learning control mostly from unlabeled play data and reduces the burden of language annotation to less than 1% of the total data. Concretely, given multiple contextual imitation datasets $\mathcal{D} = \{D^0, D^1, \ldots, D^K\}$, with a different way of describing tasks, MCIL trains a single latent goal conditioned policy $\pi_\theta\left(a_t \mid x_t, z\right)$ over all datasets simultaneously, as well as one parameterized encoder per dataset.

B. Implementation Details

We follow the baseline architecture implementation reported by Lynch *et al.* [6] unless stated otherwise. We train the agent with the Adam optimizer and a learning rate of 10^{-4}. We set the weight controlling the influence of the KL divergence to the total loss to $\beta = 0.001$. During training, we randomly sample windows between length 16 and 32 and pad them until the max length of 32. As in the original implementation, no image data augmentations are applied and absolute cartesian actions w.r.t the world frame are used. The encoder for the gripper camera takes an image of 84×84 as input and consists of 3 convolutional layers with 32, 64, and 64 channels followed by a 128 unit ReLU MLP. The encoder for the visual-tactile sensor is based on a pre-trained ResNet-18 model. The feature vectors produced by the different modality encoders are concatenated. Depth images are concatenated channel-wise with the RGB images in an early-fusion fashion. In contrast to [6], the gripper fingers of the robot in the CALVIN environment cannot be controlled independently, reducing the action output of the network by one dimension. We note that the same training hyperparameters are used for all splits.

V. Experimental Results

The results comparing language-conditioned policies based on multicontext imitation learning for the different evaluation modes in CALVIN are shown in Figure 8. We note that there is no constraint to use imitation learning approaches to solve CALVIN tasks, as approaches that use reinforcement learning to learn language-conditioned policies can also be applied [7]. We observe that the baseline with images of the static camera achieves a success rate of 53.9% for the MTLC evaluation setting, when training and testing the 34 manipulation tasks on the same environment. The success rate stays comparable when including a gripper camera, depth channels or tactile sensing. We hypothesize that the reason for not seeing larger improvements when adding the gripper camera is that the policy might benefit from using relative actions instead of

Input					Train → Test	MTLC	LH-MTLC				
Static Camera		Gripper Camera		Tactile		(34 tasks)	No. Instructions in a Row (1000 chains)				
RGB	D	RGB	D	RGB			1	2	3	4	5
✔	✗	✗	✗	✗	D → D	53.9%	48.9%	12.9%	2.6%	0.5%	0.08%
✔	✗	✗	✗	✗	A,B,C,D → D	35.6%	28.2%	2.5%	0.3%	0%	0%
✔	✗	✗	✗	✗	A,B,C → D	38.6%	20.2%	0.2%	0%	0%	0%
✔	✗	✔	✗	✗	D → D	51.8%	34.4%	5.8%	1.1%	0.2%	0.08%
✔	✗	✔	✗	✗	A,B,C,D → D	49.7%	37.3%	2.7%	0.17%	0%	0%
✔	✗	✔	✗	✗	A,B,C → D	38.0%	30.4%	1.3%	0.17%	0%	0%
✔	✗	✗	✗	✔	D → D	54.2%	28.5%	3.2%	0%	0%	0%
✔	✗	✗	✗	✔	A,B,C,D → D	47.9%	22.7%	2.3%	0.3%	0%	0%
✔	✗	✗	✗	✔	A,B,C → D	43.7%	17.3%	0.8%	0.08%	0%	0%
✔	✔	✔	✔	✗	D → D	46.1%	28.2%	4.6%	0.3%	0.08%	0%
✔	✔	✔	✔	✗	A,B,C,D → D	40.7%	14.4%	1.8%	0.08%	0.08%	0%
✔	✔	✔	✔	✗	A,B,C → D	30.8%	21.1%	1.3%	0%	0%	0%

Fig. 8: Baseline performance of MCIL [6] on the CALVIN Challenge for different combinations of training and test environments and sensor suites.

global actions. A qualitative analysis indicates that the performance depends significantly on the initial position of the robot, suggesting the agent relies on context rather than learning to disentangle initial states and tasks. It is possible this is due to causal confusion between the proprioceptive information and the target actions [32]. Besides, we did not use image data augmentations in the baselines to stay close to the original implementation, but we hypothesize this might be beneficial. Additionally, more elaborate sensor fusion approaches such as mixture of experts [33], [34] or view-invariant contrastive learning [35], [36] might be necessary to learn better multi-modal state representations.

For the Long-Horizon MTLC evaluation we observe that the agents perform poorly on CALVIN's long-horizon tasks with high-dimensional state spaces. The best MCIL model achieves a success rate of 0.08% when following chains of five language instructions in a row when training and testing on the same environment. Additionally, it solves the first subtask of the chain, starting from a neutral position, in 48.9% of the cases. We observe that the policy sometimes correctly executes block manipulation tasks, but confuses the red and blue block colors in the instruction. As the language models embed sentences containing the words red and blue similarly, backpropagating through the entire language model and leveraging auxiliary losses that try to align visual and language representations [37] might be beneficial to tackle the complicated perceptual grounding problem.

Finally, the general performance drops significantly when evaluating on the multi environment and zero-shot multi environment settings, which do not follow the standard assumption of imitation learning that training and test tasks are drawn independently from the same distribution. In order to achieve better zero-shot generalization capabilities, additional techniques from the domain adaptation literature [36], better data augmentation and a stronger focus on depth inputs, since they are invariant to texture changes, might be helpful. As MCIL is an offline learning method, we hypothesize that naïve data sharing between multiple domains can be brittle because it can exacerbate the distribution shift between the policy represented in the data and the policy being learned [38]. This motivates further research into agents that can perform the complex long-horizon language-conditioned manipulation tasks introduced by CALVIN.

VI. Conclusion

In this paper, we presented CALVIN, the first public benchmark of instruction following that combines natural language conditioning, multimodal high-dimensional inputs, 7-DOF continuous control, and long-horizon robotic object manipulation in both seen and unseen environments. As the field of language-driven robotics evolves, a need arises to standardize research for better benchmarks and more reproducible results. CALVIN has the goal of providing researchers with a modular framework that has been developed from the ground up to support training, prototyping, and validation of language-conditioned continuous control policies. Further to that, we hope, along with the help of the community, to continuously expand the tasks available for both training and evaluation.

We use CALVIN to evaluate a conditional sequence-to-sequence variational autoencoder, shown to be effective in other long horizon language-conditioned manipulation tasks [6]. While this model is relatively competent at accomplishing some subgoals, the overall success rates are poor. The long horizon of CALVIN tasks poses a significant challenge with sub-problems including the acquisition of a diverse repertoire of general-purpose skills, object detection, referring expression and action grounding, and task-agnostic continuous control. We hope CALVIN will open the door for the future development of agents that can relate human language to their perception and actions and generalize abstract concepts to unseen entities in the same way humans do.

Acknowledgement

This work was supported by the German Federal Ministry of Education and Research under contract 01IS18040B-OML. We thank Corey Lynch and Pierre Sermanet for help with the MCIL baseline.

REFERENCES

[1] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *CoRL*, 2019.

[2] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *CoRL*, 2019.

[3] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman, "Scaling up multi-task robotic reinforcement learning," in *CoRL*, 2021.

[4] S. Young, J. Pari, P. Abbeel, and L. Pinto, "Playful interactions for representation learning," *arXiv preprint arXiv:2107.09046*, 2021.

[5] S. Wang, M. Lambeta, P.-W. Chou, and R. Calandra, "Tacto: A fast, flexible and open-source simulator for high-resolution vision-based tactile sensors," *arXiv preprint arXiv:2012.08456*, 2020.

[6] C. Lynch and P. Sermanet, "Language conditioned imitation learning over unstructured data," in *RSS*, 2021.

[7] S. Nair, E. Mitchell, K. Chen, B. Ichter, S. Savarese, and C. Finn, "Learning language-conditioned robot behavior from offline data and crowd-sourced annotation," in *CoRL*, 2021.

[8] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek, "Robots that use language," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 25–55, 2020.

[9] S. Kazemzadeh, V. Ordonez, M. Matten, and T. Berg, "Referitgame: Referring to objects in photographs of natural scenes," in *EMNLP*, 2014.

[10] J. Lu, D. Batra, D. Parikh, and S. Lee, "Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks," in *NeurIPS*, 2019.

[11] R. Paul, J. Arkin, N. Roy, and T. M Howard, "Efficient grounding of abstract spatial concepts for natural language interaction with robot manipulators," in *RSS*, 2016.

[12] M. Shridhar and D. Hsu, "Interactive visual grounding of referring expressions for human-robot interaction," in *RSS*, 2018.

[13] J. Hatori, Y. Kikuchi, S. Kobayashi, K. Takahashi, Y. Tsuboi, Y. Unno, W. Ko, and J. Tan, "Interactively picking real-world objects with unconstrained spoken language instructions," in *ICRA*, 2018.

[14] T. Nguyen, N. Gopalan, R. Patel, M. Corsaro, E. Pavlick, and S. Tellex, "Robot object retrieval with contextual natural language queries," in *RSS*, 2020.

[15] H. Zhang, Y. Lu, C. Yu, D. Hsu, X. La, and N. Zheng, "Invigorate: Interactive visual grounding and grasping in clutter," in *RSS*, 2021.

[16] O. Mees and W. Burgard, "Composing pick-and-place tasks by grounding language," in *ISER*, 2021.

[17] S. G. Venkatesh, A. Biswas, R. Upadrashta, V. Srinivasan, P. Talukdar, and B. Amrutur, "Spatial reasoning from natural language instructions for robot manipulation," in *ICRA*, 2021.

[18] W. Liu, C. Paxton, T. Hermans, and D. Fox, "Structformer: Learning spatial structure for language-guided semantic rearrangement of novel objects," *arXiv preprint arXiv:2110.10189*, 2021.

[19] D. Misra, J. Langford, and Y. Artzi, "Mapping instructions and visual observations to actions with reinforcement learning," in *EMNLP*, 2017.

[20] H. Yu, H. Zhang, and W. Xu, "Interactive grounded language acquisition and generalization in a 2d world," in *ICLR*, 2018.

[21] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. Van Den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *CVPR*, 2018.

[22] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," in *CVPR*, 2020.

[23] M. Shridhar, L. Manuelli, and D. Fox, "Cliport: What and where pathways for robotic manipulation," in *CoRL*, 2021.

[24] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. B. Amor, "Language-conditioned imitation learning for robot manipulation tasks," in *NeurIPS*, 2020.

[25] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn, "Bc-0: Zero-shot task generalization with robotic imitation learning," in *CoRL*, 2021.

[26] L. Shao, T. Migimatsu, Q. Zhang, K. Yang, and J. Bohg, "Concept2robot: Learning manipulation concepts from instructions and human demonstrations," in *RSS*, 2020.

[27] V. Blukis, D. Misra, R. A. Knepper, and Y. Artzi, "Mapping navigation instructions to continuous control actions with position-visitation prediction," in *CoRL*, 2018.

[28] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021.

[29] L. P. Kaelbling, "Learning to achieve goals," in *IJCAI*. Citeseer, 1993, pp. 1094–1099.

[30] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba, "Hindsight experience replay," in *NeurIPS*, 2017.

[31] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," in *NeurIPS*, 2020.

[32] P. de Haan, D. Jayaraman, and S. Levine, "Causal confusion in imitation learning," *NeurIPS*, 2019.

[33] O. Mees, A. Eitel, and W. Burgard, "Choosing smartly: Adaptive multimodal fusion for object detection in changing environments," in *IROS*, 2016.

[34] M. A. Lee, Y. Zhu, K. Srinivasan, P. Shah, S. Savarese, L. Fei-Fei, A. Garg, and J. Bohg, "Making sense of vision and touch: Self-supervised learning of multimodal representations for contact-rich tasks," in *ICRA*, 2019.

[35] Y. Tian, D. Krishnan, and P. Isola, "Contrastive multiview coding," in *ECCV*, 2020.

[36] O. Mees, M. Merklinger, G. Kalweit, and W. Burgard, "Adversarial skill networks: Unsupervised robot skill learning from videos," in *ICRA*, 2020.

[37] 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," *arXiv preprint arXiv:2103.00020*, 2021.

[38] T. Yu, A. Kumar, Y. Chebotar, K. Hausman, S. Levine, and C. Finn, "Conservative data sharing for multi-task offline reinforcement learning," in *NeurIPS*, 2021.

APPENDIX

A. Tasks

All tasks are defined in terms of change in the environment state between the first and the final frame of a sequence. In order to see if a task was solved in an arbitrary sequence of frames of the CALVIN dataset, the environment is reset to the state of the first and the last frame of that sequence. The tasks detector compares the two simulator states and checks which task conditions are fulfilled. A key advantage of this strategy is that it enables efficient evaluation of sequences for task completion independent of their length. Figure 9 shows a list of all task definitions.

Task	**Condition**
Rotate red block right Rotate blue block right Rotate pink block right	The object has to be rotated clockwise more than 60° around the z-axis while not being rotated for more than 30° around the x or y-axis.
Rotate red block left Rotate blue block left Rotate pink block left	The object has to be rotated counterclockwise more than 60° around the z-axis while not being rotated for more than 30° around the x or y-axis.
Push red block right Push blue block right Push pink block right	The object has to move more than 10 cm to the right while having surface contact in both frames
Push red block left Push blue block left Push pink block left	The object has to move more than 10 cm to the left while having surface contact in both frames
Move slider left	The sliding door has to be pushed at least 12 cm to the left.
Move slider right	The sliding door has to be pushed at least 12 cm to the right.
Open drawer	The drawer has to pulled out at least 10 cm.
Close drawer	The drawer has to be pushed in at least 10 cm.
Lift red block table Lift blue block table Lift pink block table	The object has to be grasped from the table surface and lifted at least 5 cm high. In the first frame the gripper may not touch the object.
Lift red block slider Lift blue block slider Lift pink block slider	The object has to be grasped from the surface of the sliding cabinet and lifted at least 3 cm high. In the first frame the gripper may not touch the object.
Lift red block drawer Lift blue block drawer Lift pink block drawer	The object has to be grasped from the surface of the drawer and lifted at least 5 cm high. In the first frame the gripper may not touch the object.
Place in slider	The object has to be placed in the sliding cabinet. It must be lifted by the gripper in the first frame.
Place in drawer	The object has to be placed in the drawer. It must be lifted by the gripper in the first frame.
Push into drawer	The object has to be pushed into the drawer. It has to touch the table surface in the first frame.
Stack blocks	A block has to be placed on top of another block. It may not be in contact with the gripper in the final frame.
Unstack blocks	A block that is stacked on another block has to be removed from the top, either by grasping it or by pushing it down. It may not be in contact with the gripper in the first frame.
Turn on light bulb	The switch has to be pushed down to turn on the yellow light bulb.
Turn off light bulb	The switch has to be pushed up to turn off the yellow light bulb.
Turn on LED	The button has to be pressed to turn on the green LED light.
Turn off LED	The button has to be pressed to turn off the green LED light.

Fig. 9: List of all 34 tasks with their respective success criteria.

B. Language Annotation Generation

The language annotations are extracted automatically from the recorded data with the following procedure: we randomly sample sequences with a window size of 64 frames. For each sequence the task detector checks if a task has been solved between the first and the last frame. Additionally, we check that neither that task nor any other task is solved in the first half of the sequence. The intuition behind this is that we want to include the locomotion behavior prior to the actual task. For example, before opening the drawer, the arm must navigate in the direction of the handle. This is important for learning to solve tasks with language goals from arbitrary starting positions. If a sequence qualifies for labeling, we sample a natural language instruction from a set of predefined sentences with approximately 11 synonymous instructions per task. In total, this gives 389 unique language instructions for 34 tasks. The sequence in which the task "stack blocks" is solved could for example get instructions such as "place the grasped block on top of another block" or "stack blocks on top of each other". In order to simulate a real-world scenario where it might not be possible to pair all the collected robot experience with crowd-sourced language annotations, we annotate only 1% of the recorded robot interaction data with language instructions. The CALVIN dataset conveniently includes precomputed MiniLM language embeddings for all instructions, but researchers are free to use their own language model of choice on the raw input data.