Learning Language-Conditioned Robot Behavior from Offline Data and Crowd-Sourced Annotation

Suraj Nair[1], **Eric Mitchell**[1], **Kevin Chen**[1], **Brian Ichter**[2], **Silvio Savarese**[1], **Chelsea Finn**[1,2]
[1]Stanford University, [2]Robotics at Google

Abstract: We study the problem of learning a range of vision-based manipulation tasks from a large offline dataset of robot interaction. In order to accomplish this, humans need easy and effective ways of specifying tasks to the robot. Goal images are one popular form of task specification, as they are already grounded in the robot's observation space. However, goal images also have a number of drawbacks: they are inconvenient for humans to provide, they can over-specify the desired behavior leading to a sparse reward signal, or under-specify task information in the case of non-goal reaching tasks. Natural language provides a convenient and flexible alternative for task specification, but comes with the challenge of grounding language in the robot's observation space. To scalably learn this grounding we propose to leverage *offline robot datasets* (including highly sub-optimal, autonomously collected data) with *crowd-sourced natural language labels*. With this data, we learn a simple classifier which predicts if a change in state completes a language instruction. This provides a language-conditioned reward function that can then be used for offline multi-task RL. In our experiments, we find that on language-conditioned manipulation tasks our approach outperforms both goal-image specifications and language conditioned imitation techniques by more than 25%, and is able to perform visuomotor tasks from natural language, such as "open the right drawer" and "move the stapler", on a Franka Emika Panda robot.

Keywords: Natural Language, Offline RL, Visuomotor Manipulation

1 Introduction

We are motivated by the goal of generalist robots which can be commanded to complete a diverse range of manipulation tasks. Doing so requires humans to be able to effectively specify tasks for the robot to solve. One popular approach to task specification is through goal-states, which by definition are grounded in the robot's observation space, making them a natural choice for self-supervised techniques [1, 2, 3, 4]. However, goal-state specification comes with a number of drawbacks, including (a) human effort required in generating a goal state to provide the robot, (b) task over-specification resulting in a sparse reward signal (e.g. a goal image for the task of pushing a *single* object also specifies positions of all other objects and the robot itself), and (c) task under-specification (e.g. moving to the right indefinitely). Natural language presents a promising alternative form of specification, providing an easy way for humans to communicate tasks. Moreover, natural language can flexibly represent non-goal reaching tasks and tasks with varying degrees of specificity,

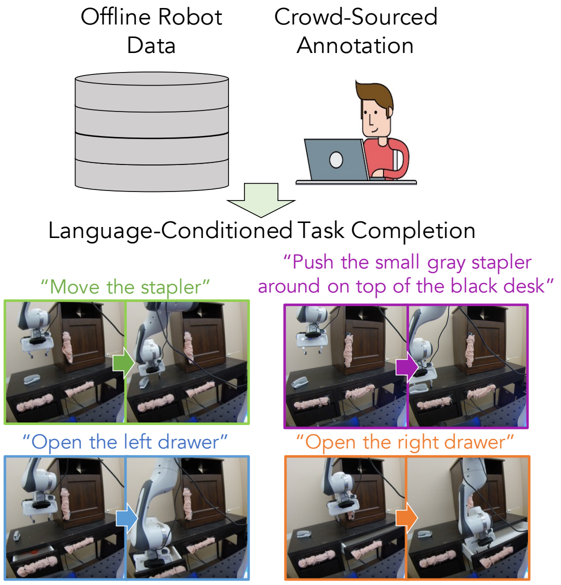


Figure 1: We learn language-conditioned visuomotor policies using sub-optimal offline data, crowd-sourced annotation, and pre-trained language models, enabling a real robot to complete language-specified tasks while being robust to complex rephrasings of the task description.

such as grasping *any* marker from a desk with several markers, while a single goal image could only capture one instance of success. To this end, we study the problem of learning *language-conditioned visuomotor manipulation skills from offline datasets of robotic interaction.*

Despite the abundant benefits of being able to command robots with natural language, such agents have remained out of reach. A major challenge in acquiring such agents is that the language instructions need to be grounded in the agent's high-dimensional observation space. Learning this grounding is difficult, and requires diverse interaction data paired with language annotations. Recent works have made progress towards learning such grounding by annotating data collected by humans [5, 6, 7]; however, collecting many human teleoperated trajectories on real robots can be costly and time consuming, and is thus difficult to scale to a broad set of language conditioned behaviors.

Our key insight is that a practical and scalable way to ground language is to combine autonomously-collected offline datasets of robotic interaction with post-hoc crowd-sourced natural language labels. Unlike prior work, we do not assume this data comes from a human expert or contains optimal actions, allowing the agent to leverage a wide range of data sources such as autonomous exploration data (e.g. random, scripted, intrinsically motivated), replay buffers of trained RL agents, human expert data (e.g. demonstrations, human play), and data without action labels. Given such pre-collected data, we can then use crowd-sourcing to scalably label trajectories with natural language labels describing the behaviors in the data. To learn from this sub-optimal data with noisy annotations, we learn a classifier which takes as input a natural language instruction and an initial and final image, and predicts whether or not the transition completes the instruction. This learned classifier can then be used as a language-conditioned reward for offline RL to learn language-conditioned behaviors.

Concretely, in this work we propose to learn language conditioned skills from vision using sub-optimal, autonomously-collected offline data and crowd-sourced annotations (See Figure 1). We present a simple technique to learn language-conditioned rewards from this data, which we call **L**anguage-conditioned **O**ffline **R**eward **L**earning (LORL), and combine it with visual model-predictive control to complete language conditioned tasks (See Figure 2). In our experiments in simulation, we observe that even with data collected by a random policy our proposed method solves language-conditioned object manipulation tasks 25% more effectively than language conditioned imitation learning techniques, as well as $\sim 30\%$ more effectively than goal-image conditioned comparisons which over-specify the task. Additionally, we observe that by virtue of leveraging pretrained language models our learned reward is capable of generalizing from scripted language instructions to unseen natural language zero-shot, suggesting that knowledge in pretrained language models can enable more efficient learning of grounded language as observed in prior work [6, 8]. Finally, we leverage an existing real robot dataset of sub-optimal data, label the dataset using crowd-sourcing, and use it to complete five visuomotor tasks specified by natural language, such as "open the right drawer" or "move the stapler" on a real Franka Emika Panda robot.

2 Related Work

There is a rich literature of work which studies interactive agents, and grounding their behaviors in language [9, 10, 11, 12]. Many prior works have studied this problem in the context of *instruction following*, where an agent aims to complete a task specified by formal language/programs [13, 14, 15, 16, 17, 18] or natural language [10, 11, 19, 20]. While these approaches have been largely studied in simulated spatial games [19, 21, 22, 23] or in object-directed visual navigation in simulated robots [24, 25, 26, 27, 28, 29, 23] some of which include high-level object interaction [30], in this work we focus on the domain of learning control for *vision-based robotic manipulation*.

Early works have approached instruction following with strategies like semantic parsing mapped to motion primitives or pre-defined actions to execute tasks in virtual domains [31, 32, 33, 34] and on mobile robots [35, 36]. Like our approach these methods don't require expert demonstrations; however unlike these approaches, we directly learn robotic control from images and natural language instructions, and don't assume any predefined motion primitives. More recently, end-to-end deep learning has been used to condition agents on natural language instructions [37, 24, 38, 27, 39, 6, 7, 40], which are then trained under an imitation and/or reinforcement learning objective. In the reinforcement learning setting, works have adopted a range of strategies, from language-conditioned reinforcement learning while leveraging environment rewards [21, 41, 42, 43, 44, 45] to using language as a reward bonus to densify the environment reward and aid in exploration [46, 47, 48, 49, 50]. In contrast, we do not assume any environment provided reward signal, and rather aim to *learn* effective language-conditioned rewards from annotated data of interaction.

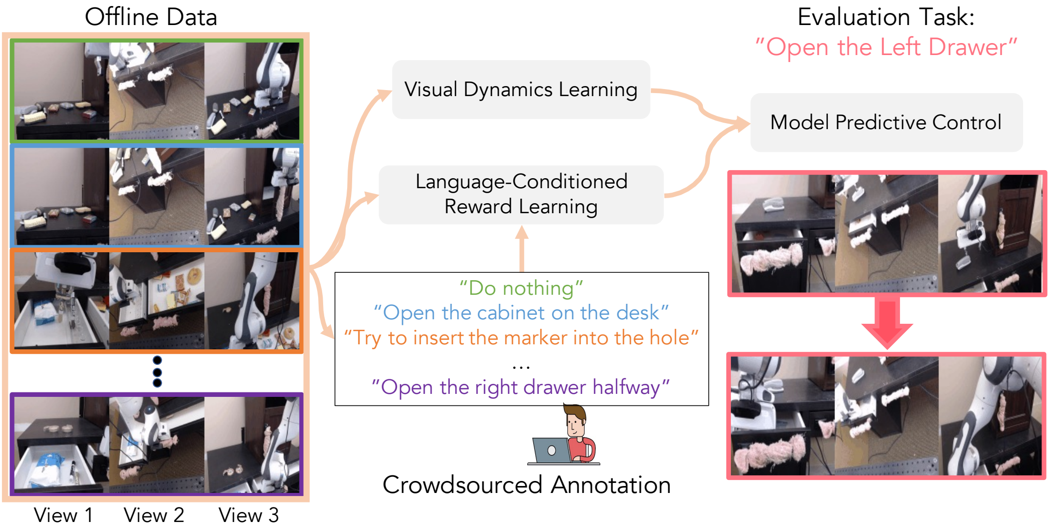


Figure 2: **Language-conditioned Offline Reward Learning (LORL)**. We propose a technique to learn language-conditioned behavior from offline datasets of robot interaction (**left**). To do so, we crowd-source natural language annotations describing the behavior in the offline data, and use it to learn a language-conditioned reward function (**middle**). We then combine this reward and a learned visual dynamics model through model predictive control to complete language specified tasks form vision on a real robot (**right**).

Numerous prior works have also studied learning language conditioned rewards online from demonstrations or examples of successful completion of tasks and language annotations [5, 22, 39, 49]. These works then use the learned reward to optimize policies through online RL, often using the agents own online experience to train the reward [22, 51]. While these works also learn language-conditioned rewards, and in some cases also use discriminative techniques to learn the reward [22, 49], running language-conditioned online RL on a physical robot can be prohibitively time consuming. Our work aims to learn language-conditioned behaviors from entirely offline datasets (which may be highly sub-optimal), making it feasible to learn language-conditioned behaviors on real robots.

Other works have studied using offline data in the form of demonstrations [7] or human teleoperated trajectories (i.e. "play data") [6], to learn language-conditioned robotic agents in simulation. Most related is Lynch and Sermanet [6] who also use crowd-sourcing to annotate play data with natural language instructions. Critically, these works treat the offline data as near optimal, to the extent that behavior cloning techniques can be used to learn language-conditioned policies. Unlike these works, we don't make any assumptions about the optimality of the actions in the collected data, allowing the agent to learn from broader offline datasets, including autonomously collected data, which can be considerably easier to collect at scale on a real robot than human tele-operation data [52, 53]. Moreover, we observe in Section 5.1 that our proposed approach outperforms imitation learning techniques on such data, and in Section 5.3 that our method is effective on a real robot.

Many prior works have studied how robots can learn to complete a wide range of tasks from vision. While many approaches have been taken to task-specification, including task IDs [54, 55], robot and human demonstrations [56, 57, 58], and meta-learning from rewards [59], a common approach is goal-conditioned learning [60, 61, 2, 1], where an agent learns to reach particular goal states or distributions [62]. Many approaches have been applied to this domain, ranging from goal-conditioned model-free learning [2, 63, 55, 64] with goal relabeling [61], model-based planning with a learned visual dynamics model [65, 66], to methods which combine the both [67]. Unlike these works, the focus of this work is multi-task visuomotor learning from natural language specifications. Furthermore, we find in Section 5.1 that using our language-conditioned reward we can more effectively complete tasks than leveraging a goal image specification, while requiring less human effort to specify the task.

3 Preliminaries

In this work we consider an interactive agent which aims to complete K tasks $\{\mathcal{T}_k\}_1^K \subset \mathcal{T}$ where $\mathcal{T}$ denotes the space of all tasks. For each task $\mathcal{T}_i \in \mathcal{T}$ the agent operates in a Markov decision process MDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, p, \mathcal{R}_i, T)$ where $\mathcal{S}$ is the state space (in our case RGB images), $\mathcal{A}$ is the robot's action space, $p(s_{t+1}|s_t, a_t)$ is the robot environment's stochastic dynamics, $\mathcal{R}_i : \mathcal{S} \times \mathcal{S} \rightarrow \{0, 1\}$

indicates the binary reward at state s for completing task $\mathcal{T}_i$ from initial state s_0, and T is the episode horizon. Lastly, let $\mathcal{L}$ denote the set of all natural language, and let $\mathcal{L}_i \subset \mathcal{L}$ denote the set of language instructions which describe task i. Note that there can exist many instructions $l \in \mathcal{L}_i$ which describe a task $\mathcal{T}_i$ (e.g. "pick up the blue marker" and "grasp and lift the blue marker"), and any particular instruction $l \in \mathcal{L}$ can describe multiple tasks (e.g. "pick up the marker" can describe the task of picking up the blue marker or the green marker).

In this work we assume that the true reward function $\mathcal{R}_i$ for each task $\mathcal{T}_i$ is unobserved, and must be inferred from natural language. Concretely, we assume access to an offline dataset $\mathcal{D}$ of N trajectories, where each trajectory τ_n is a tuple containing a sequence of states and actions, and a single natural language instruction $\tau_n = ([(s_0, a_0), (s_1, a_1), ..., (s_T)], l_n)$. We assume that $l_n \in \mathcal{L}_i$ for at least one task $\mathcal{T}_i \in \mathcal{T}$ for which $\mathcal{R}_i(s_0, s_T) = 1$. Note $\mathcal{T}_i$ does not need to be in $\{\mathcal{T}_k\}_1^K$, meaning the offline data/annotations can consist of tasks unrelated to the robot's target tasks (e.g. "doing nothing"). Our goal then is to learn a parametrized reward model $\mathcal{R}_\theta : \mathcal{S} \times \mathcal{S} \times \mathcal{L} \rightarrow [0, 1]$ which conditioned on a language instruction l, initial state s_0, and state s infers the true reward function $\mathcal{R}_i(s_0, s)$ for *some* task $\mathcal{T}_i$ for which $l \in \mathcal{L}_i$. Given $\mathcal{R}_\theta$, we aim to instantiate a stochastic language-conditioned policy $\pi : \mathcal{S} \times \mathcal{S} \times L \rightarrow \mathcal{A}$, which conditioned on a natural language instruction l produces actions to maximize the expected sum of rewards $\sum_{t=0}^{T} \mathcal{R}_i(s_0, s_t)$ for the inferred task $\mathcal{T}_i$. Note that this formulation captures tasks that are reflected in a change of state, but not tasks which are path dependent (e.g. "close the drawer *slowly*").

4 Language-conditioned Offline Reward Learning (LORL)

Now we describe how we go about learning our parametrized language-conditioned reward $\mathcal{R}_\theta$ from $\mathcal{D}$, as well as how we instantiate our language-conditioned policy π to maximize the learned reward, also shown in Figure 2. Our key idea is that while we cannot make assumptions about the optimality of the behavior in $\mathcal{D}$, we can use the the initial and final states of each trajectory and provided language annotations to ground what changes in state correspond to successful completion of language instructions. Then to learn control we can leverage all of the data in $\mathcal{D}$ to learn a global task-agnostic model of the dynamics of the robots environment, which can be combined with the learned reward via model-predictive control (MPC) to complete language conditioned tasks.

4.1 Learning the Reward Function

Given the provided dataset $\mathcal{D} = [\tau_1, ..., \tau_N]$ of N trajectories $\tau_n = ([(s_0, a_0), (s_1, a_1), ..., (s_T)], l_n)$, how might we go about learning our reward function $\mathcal{R}_\theta$? Critically, the behavior policy which collected this data could be sub-optimal. Therefore, we cannot assume the optimality of any particular action taken. However, because the human provided annotations describe the task being completed in the video, the assumption we can make about the data is that going from the start to the end of the trajectory constitutes completion of l_n. Therefore, we implement our reward function as a binary classifier $\mathcal{R}_\theta(s_0, s, l)$ which looks at the initial state s_0, current state s, and language instruction l and predicts if going from the initial state to the current state satisfies the language instruction.

Training the reward function in this manner has numerous favorable properties. First, unlike explicitly predicting a single instance of a successful goal state for a language instruction or vice-versa, a classifier can easily capture the many-to-many mapping that exists between language instructions and tasks. Doing so allows it to capture the full space of successful behavior even in cases where there exists many possible language instructions l which can describe completing a task $\mathcal{T}_i$ and many possible pairs of initial and final states (s_0, s_T) which can constitute successfully completing any given instruction l. Second, by virtue of being *context-dependent* on the initial state, the reward function can be used in closed loop planning to perform behaviors indefinitely without additional specification (i.e. the reward for "move right" is relative to the agent's current position, so applying it iteratively will encourage the agent to continuously move right). Lastly, unlike other works which use classifiers for single-task reward learning [68, 69, 53] on robots, a language-conditioned reward classifier can flexibly represent many tasks with an easy to provide form of task-specification. Concretely, we sample positive examples $(s_0, s_T, l_t) \in \tau_n \sim \mathcal{D}$ from the annotated dataset which constitute successfully completing an instruction and generate negative examples which don't complete instructions $(s'_0, s'_T, l') \sim \mathcal{N}$ also from $\mathcal{D}$ (described in detail in the next sections). We then minimize the binary cross entropy loss:

$$\mathcal{J}(\theta) = \mathbb{E}_{(s_0, s_T, l) \sim \mathcal{D}}[\log(\mathcal{R}_\theta(s_0, s_T, l))) + \mathbb{E}_{(s'_0, s'_T, l') \sim \mathcal{N}}[\log(1 - \mathcal{R}_\theta(s'_0, s'_T, l'))]. \quad (1)$$

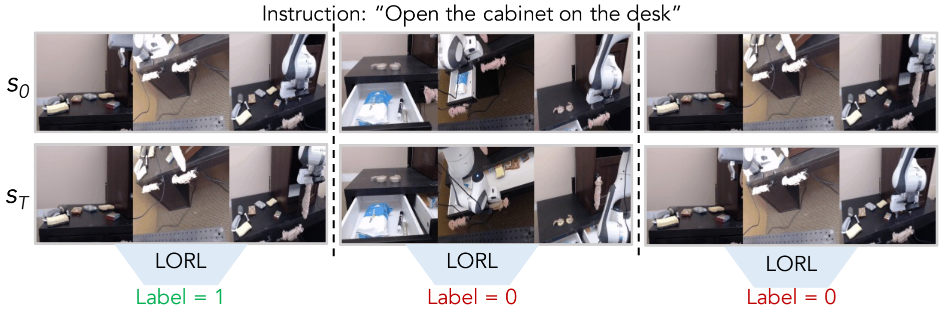


Figure 3: **Training LORL.** We train LORL on balanced batches of positive examples where the initial/final image transition satisfies the language command (**left**), negative examples where the initial/final states satisfy a *different* instruction (**middle**), and negative examples where the initial and final image are reversed (**right**).

Positive Selection. Selecting positive examples for the classifier is straightforward, as we know the initial and final state in an episode labeled with instruction l satisfy that command. However, it is also highly likely that there are other states near the beginning and end of the episode for which the instruction is satisfied. Therefore we employ a noisy labeling scheme where we label any (s_i, s_j, l) for which $i \leq \alpha T$ and $j \geq (1-\alpha)T$ as a positive. Higher values of α may occasionally include false positives, but also significantly increase the set of positives which can be used for training.

Negative Selection. First, we choose initial and final states from episodes with *different* language instructions as negatives. Specifically, for language command l, we select negatives s'_0, s'_T by selecting any $(s_0, s_T, l') \sim \mathcal{D}$ where $l' \neq l$. Note that since there may be instructions $l' \neq l$ which describe the same task, this may occasionally yield false negatives, however like prior work [68, 53] we find that we can learn an effective reward despite noisy negatives. Second, to encourage the reward to capture temporal progress (as opposed to focusing on spurious visual features), we also include the example (s_T, s_0, l) as a negative in training $\mathcal{R}_\theta$. Ultimately, we train on balanced batches of positive examples and both types of negative examples (See Figure 3).

Data Augmentation. Reward functions trained using classifiers have been shown to be prone to over-fitting creating a sparse or incorrect reward signal [53]. This issue is further exacerbated by the fact that we only have limited positive examples per episode. To combat this, we use visual data augmentation in the form of affine transformations and color jitter as well as uniform noise in the embedding space of language instructions to prevent classifier over-fitting.

Leveraging Pre-Trained Language Models. Finally, learning the meaning of raw natural language while simultaneously grounding the robot's actions using only crowd-sourced data of a few thousand robot episodes with language annotations poses a significant challenge. Therefore we leverage a fixed pre-trained distilBERT sentence encoder [70], to encode the natural language commands into a fixed length vector in $\mathbb{R}^{768}$ before they go into the classifier. We find in Section 5.2 that by using the pre-trained model we can generalize to unseen natural language commands from synthetic data.

4.2 Learning Language Conditioned Policies with Visual Model Predictive Control

Once trained, the learned reward function $\mathcal{R}_\theta$ in principle could be used with any form of offline reinforcement learning to learn language-conditioned policies. In this work, we aim to learn visuomotor control on real robots from large datasets of sub-optimal or even random offline data. Model-based RL techniques have been particularly effective in this endeavor [65, 58], and in our case all offline data can be used to train a single task-agnostic visual dynamics model. We then use this model with planning to maximize the learned language-conditioned reward $\mathcal{R}_\theta$. Specifically, we learn a forward visual dynamics model $s_{t+1} \sim p_\theta(s_t, a_t)$ which is trained on the entire offline dataset $\mathcal{D}$, and does not use language annotations. We leverage off-the-shelf action-conditioned video prediction frameworks for learning this model [71, 72], which we describe in detail in the supplement.

Given the learned dynamics model p_θ and the learned reward function $\mathcal{R}_\theta$, we then use model predictive control to instantiate a policy to complete language-conditioned tasks. Specifically, given a language instruction l and initial state s_0, we sample M different actions sequences of length H, which we feed through p_θ to get a predicted future state $\hat{s}^m_{t+H}$. For each prediction we compute the reward as $\mathcal{R}_\theta(s_0, \hat{s}^m_{t+H}, l)$. Action sequences are optimized to maximize reward using the cross-entropy method (CEM) [73], until the best action sequence is applied in the environment (Figure 4).

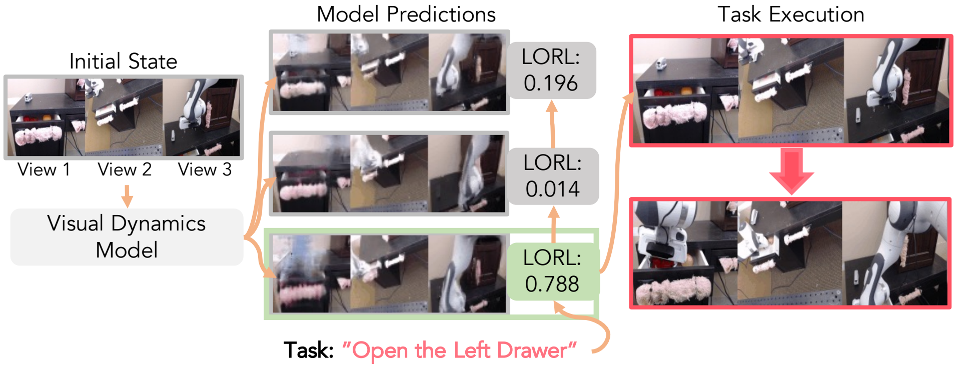


Figure 4: **Executing Language-Conditioned Policies with LORL.** To execute language-conditioned behavior, we perform model predictive control with a learned visual dynamics model and LORL. Specifically, from the initial state we predict many future states for different action sequences (**left/middle**). We then rank those sequences according to the LORL reward for the user specified natural language instruction (**middle**). After multiple iterations, the best action sequence is stepped in the environment executing the task (**right**).

5 Experiments

In our experiments we aim to study three main questions. **(1)** How does our proposed method for learning language conditioned policies from offline data compare to both language-conditioned and goal-image conditioned prior methods? **(2)** By virtue of using pre-trained language models, to what extent can our method generalize to unseen natural language commands? **(3)** Can our method be used to solve visuomotor tasks on a real robot using crowd-sourced annotations? We study experiments (1) and (2) in simulation, and experiment (3) on a Franka Emika Panda robot positioned in front of a desk. For qualitative results and videos, please see `https://sites.google.com/view/robotlorl`.

Simulated Domain

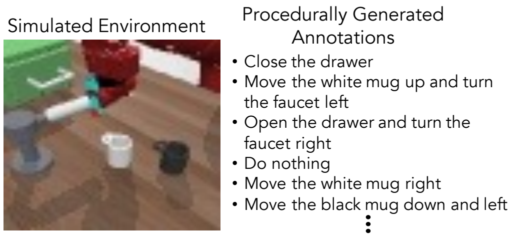

Simulated Environment
Procedurally Generated Annotations
• Close the drawer
• Move the white mug up and turn the faucet left
• Open the drawer and turn the faucet right
• Do nothing
• Move the white mug right
• Move the black mug down and left
⋮

Figure 5: **Simulated Domain/Data.** We leverage a simulated domain developed on top of Meta-World [54] which contains a Saywer robot interacting with a drawer, faucet, and two mugs (**left**). We collect data using a random policy, and annotate episodes with language instructions using the environment state (**right**).

We study our first two experimental questions in a simulated domain developed on top of the Meta-World [54] environment, where a simulated sawyer robot interacts on a tabletop with a drawer, a faucet, and two mugs (see Figure 5 (left)). In this domain, we collect an offline dataset of 50,000 episodes by running a random policy in the environment, and label episodes procedurally using the true environment state yielding 2311 unique instructions (see Figure 5 (right)). After training on this data, we evaluate on 6 seen tasks which involve (1) closing the drawer, (2) opening the drawer, (3) turning the faucet left and (4) right, and (5) pushing the black mug right, and (6) pushing the white mug down.

5.1 Does LORL enable effective language-conditioned behavior compared to prior work?

In this experiment we aim to evaluate how LORL compares to prior techniques for learning language and goal image conditioned behavior on the 6 target tasks described previously.

Comparisons. We compare LORL (**Ours**) to language-conditioned behavior-cloning (**LCBC**), which imitates the behavior in the offline dataset conditioned on the language instruction label, which is reflective of prior works that use imitation learning to learn language-conditioned behavior [6, 7]. We also compare to language-conditioned RL (**LCRL**), which labels the final state in each episode as having reward 1 for the annotated language instruction and 0 elsewhere, and trains a language conditioned-policy using offline Q learning, which reflects a fully offline version of the low-level policy used in [41]. Furthermore, we compare to using a goal-image as the task specification instead of language, and provide the agent with a ground truth goal-image of the object in its desired position, with which we use either L2 pixel distance (**Pixel**) or LPIPS [74] similarity (**LPIPS**) as a planning cost, reflective of prior work in visual MPC [65, 75]. Finally, we include an (**Oracle**) which uses the ground truth dynamics model and ground truth reward indicating the upper bound on the performance of the CEM planner, as well as the performance of a (**Random**) policy. All comparisons use the same architecture and data where possible; see the supplement for further details.

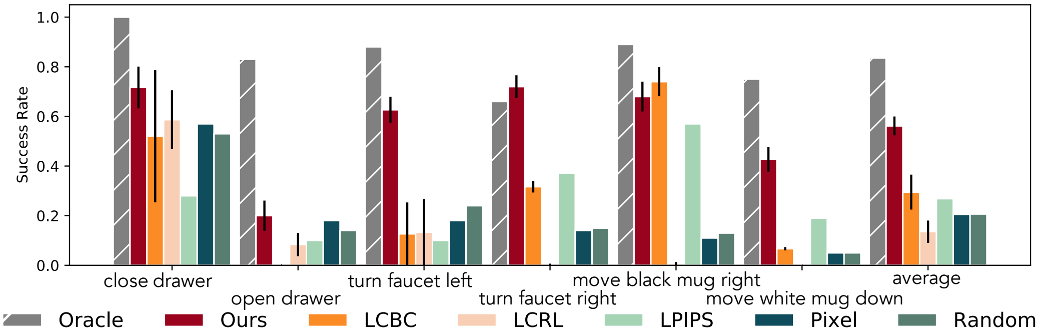


Figure 6: **Comparison to Prior Work.** On 6 simulated language-conditioned tasks, we find that LORL (Ours) outperforms language-conditioned imitation learning (LCBC) and Q-learning (LCRL) as well as goal-image task specification (LPIPS/Pixel) by over 25%. Success rates/standard error computed over 3 seeds of 100 trials.

Results. Figure 6 shows the success rates over 3 seeds of 100 trials, ordered by legend. We observe first that our proposed approach outperforms the next best method, language-conditioned behavior cloning, by more than 25%. By learning a language-conditioned reward and planning over it, the robot executes the tasks more effectively than what it observed in the data. On the other hand, because the data is sub-optimal, language-conditioned imitation learning is only able to learn coarse directions associated with each task, and as a result fails on tasks that require more fine-grained motion like "turn faucet left". Second, we observe that language-conditioned RL with a binary reward struggles to learn at all, indicating the difficulty in jointly learning language-grounding and control. Finally, we find that using goal-images with a pixel cost also fails, performing comparably with a random policy. We observe that the agent tries to match the arm position in the goal instead of the interacting with the objects, highlighting the limitation of goal-images in their tendency to *over-specify* the task. Using LPIPS similarity improves performance, however is still $\sim 30\%$ worse than our method.

5.2 Can LORL generalize zero-shot to unseen natural language commands?

In our second experiment, we study our methods ability to generalize to unseen instructions by nature of using pre-trained language models. Specifically, for the six target tasks, we test our method with a *rephrased instruction*, which was completely unseen during training. We evaluate task performance on the **Original** commands, on the commands with an **Unseen Verb** (e.g. "turn faucet left" → "rotate faucet left"), on the command with an **Unseen Noun** (e.g. "move black mug right" → "move dark cup right"), and **Unseen Verb+Noun** (e.g. "close drawer" → "shut cabinet"). Finally, we also test on **Unseen Natural Language** commands collected from 9 human volunteers who were asked to rephrase the command in a creative way, for example "turn faucet left"→ "Spin nozzle left". The full set of unseen instructions for each task is in the supplement.

Success Rate	LORL	LORL (-PM)
Original	**56 ± 1%**	40 ± 1%
Unseen Verb	**51 ± 3%**	33 ± 2%
Unseen Noun	**51 ± 1%**	39 ± 4%
Unseen Verb + Noun	**47 ± 2%**	17 ± 3%
Unseen Natural Language	**46 ± 2%**	21 ± 1%

Table 1: **Generalization to Unseen Commands.** We compare the models ability to generalize to unseen instructions and natural language when learning with a fixed, pre-trained language model (LORL), compared to training the same architecture from scratch only on the grounded language data (LORL (-PM)). We see significant performance improvement in both seen/unseen commands by using the pre-trained model.

In Table 1, we see that on average when changing the verb *or* noun, we only see a drop in success rate of 5%, and when changing both or using human provided natural language, we see at most a 10% drop in success rate. Furthermore, we compare performance with and without using the pre-trained language model, and observe that without the pre-trained language model performance is worse on seen instructions and drops significantly more on unseen instructions (up to 23% vs 10%), suggesting that the pre-trained language model is essential to learning and generalization, consistent with results in prior work on language-conditioned imitation [6]. This result also suggests that the ungrounded knowledge in large language models may enable learning language groundings from small datasets or *entirely programmatic language* which can generalize to natural language.

5.3 Can LORL be used to learn language-conditioned visuomotor skills on a real robot?

Finally, we study the efficacy of our method in learning language-conditioned behavior on a real robot using sub-optimal offline data and crowd-sourced annotation. We consider a Franka Emika

Panda robot mounted over an IKEA desk with two drawers and a cabinet, which can hold a range of objects (Figure 7). The robot's observation space consists of 4 camera viewpoints, each providing 64×64 RGB images. The robot's action space is delta end-effector control.

Data and Annotation. We use an offline 3000 episode (150000 frame) dataset without any modification from concurrent work [76], which trains policies for different behaviors on the IKEA desk using online RL. As a result, our dataset consists of diverse behaviors, but is also sub-optimal in that it comes from the replay buffer of a learning policy which will often not complete any task or will complete tasks in highly sub-optimal ways. To annotate the data, we leverage crowd-sourcing, specifically Amazon Mechanical Turk. We ask human annotators to describe the behavior, if any, that the robot is doing, and to phrase it as a command without any pre-specified template. We collect 6000 annotations, two per episode, containing a total of 1699 unique instructions, examples of which can be seen in Figure 7. We filter out episodes for which annotators wrote the robot did nothing or indicated they could not understand what the robot was doing. See supplement for details about the environment, data, tasks, distribution of annotations, and annotator interface.

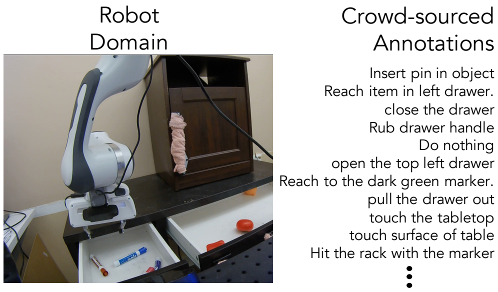


Figure 7: **Robot Domain/Data.** Our real robot domain consists of a Franka Emika Panda mounted over an IKEA desk (**left**). We crowd-source annotations describing the tasks being completed in an offline dataset from this environment (**right**).

Results. We find in Table 2 that, by training LORL on this annotated dataset and using it for visual MPC with a learned dynamics model in this domain, the robot can complete 5 language-conditioned skills with a 66% success rate on average across skills. Additionally, we find that removing negative training examples with the initial/final state flipped (**LORL (-FN)**) reduces performance by 30%, suggesting that such negatives are important for the reward to capture temporal progress and prevent over-fitting to objects.

Finally, we test LORL's robustness to more complex vocabulary and instruction length, by replacing the commands for opening the left drawer and moving the stapler with "Open the small black and white drawer on the left fully" and "Push the small gray stapler around on top of the black desk" respectively. We find that LORL is still able to succeed **7/10** and **5/10** times respectively, providing evidence that it is robust to instruction phrasing, consistent with the results in Section 5.2.

Task (10 Trials Each)	LORL	LORL (-FN)
"Open the left drawer"	**90**%	30%
"Open the right drawer"	**40**%	0%
"Move the stapler"	**50**%	0%
"Reach the marker"	**70**%	**70**%
"Reach the cabinet"	**80**%	**80**%
Average over tasks	**66**%	36%

Table 2: **Real Robot Results.** Using LORL we are able to complete 5 language-conditioned skills on a robot specified by language with a 66% success rate.

6 Limitations and Future Work

We have presented LORL, a technique for learning language conditioned behavior from offline data and crowd sourced annotation that is effective for visuomotor control on real robots and is capable of generalizing to unseen language instructions. However, a number of limitations remain. First, in its current form LORL can only capture tasks which are reflected in some change of state, but cannot capture tasks which are path dependent (e.g. "move in a circle slowly"). One exciting direction for future work to address this is to train LORL on full video clips. Second, while in this work we have focused on learning short-horizon skills from language, composing these skills to solve long-horizon language-specified tasks is important for making robots useful in the real world. Data sources with long-horizon behaviors, and more powerful planners and visual dynamics models are necessary to enabling these longer horizon tasks. Finally, while we have presented language specification as an alternative to goal images, goal images maintain the benefit of being self-supervised and in some cases can be effective task specification. Unifying both forms of specification for robots would be a valuable future direction.

Acknowledgments

The authors would like to thank Bohan Wu for valuable input throughout the project and assistance in conducting the robot experiments. The authors would also like to thank members of the IRIS and RAIL labs for providing valuable feedback. Suraj Nair is funded in part by an NSF GRFP. Eric Mitchell is funded by a Knight-Hennessy Fellowship. The authors would also like to thank the AMT annotators who helped label the robot data. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

References

[1] C. Finn and S. Levine. Deep visual foresight for planning robot motion. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[2] A. V. Nair, V. Pong, M. Dalal, S. Bahl, S. Lin, and S. Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, 2018.

[3] C. Colas, P. Fournier, O. Sigaud, and P.-Y. Oudeyer. Curious: Intrinsically motivated multi-task, multi-goal reinforcement learning. *ArXiv*, abs/1810.06284, 2018.

[4] B. Ichter, P. Sermanet, and C. Lynch. Broadly-exploring, local-policy trees for long-horizon task planning. *ArXiv*, abs/2010.06491, 2020.

[5] J. Fu, A. Balan, S. Levine, and S. Guadarrama. From language to goals: Inverse reinforcement learning for vision-based instruction following. *ArXiv*, abs/1902.07742, 2019.

[6] C. Lynch and P. Sermanet. Grounding language in play. *ArXiv*, abs/2005.07648, 2020.

[7] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. B. Amor. Language-conditioned imitation learning for robot manipulation tasks. *ArXiv*, abs/2010.12083, 2020.

[8] F. Hill, S. Mokra, N. Wong, and T. Harley. Human instruction-following with deep reinforcement learning via transfer-learning from text. *ArXiv*, abs/2005.09382, 2020.

[9] T. Winograd. Understanding natural language. *Cognitive Psychology*, 3(1):1–191, 1972. ISSN 0010-0285. doi:https://doi.org/10.1016/0010-0285(72)90002-3. URL https://www.sciencedirect.com/science/article/pii/0010028572900023.

[10] M. MacMahon, B. Stankiewicz, and B. Kuipers. Walk the talk: Connecting language, knowledge, and action in route instructions. 01 2006.

[11] T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *HRI 2010*, 2010.

[12] J. Luketina, N. Nardelli, G. Farquhar, J. N. Foerster, J. Andreas, E. Grefenstette, S. Whiteson, and T. Rocktäschel. A survey of reinforcement learning informed by natural language. In *IJCAI*, 2019.

[13] J. Andreas, D. Klein, and S. Levine. Modular multitask reinforcement learning with policy sketches. *ArXiv*, abs/1611.01796, 2017.

[14] M. Denil, S. G. Colmenarejo, S. Cabi, D. Saxton, and N. D. Freitas. Programmable agents. *ArXiv*, abs/1706.06383, 2017.

[15] S.-H. Sun, T.-L. Wu, and J. J. Lim. Program guided agent. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=BkxUvnEYDH.

[16] Y. Yang, J. Inala, O. Bastani, Y. Pu, A. Solar-Lezama, and M. Rinard. Program synthesis guided reinforcement learning. *ArXiv*, abs/2102.11137, 2021.

[17] B. Gangopadhyay, H. Soora, and P. Dasgupta. Hierarchical program-triggered reinforcement learning agents for automated driving. *ArXiv*, abs/2103.13861, 2021.

[18] E. Brooks, J. Rajendran, R. L. Lewis, and S. Singh. Reinforcement learning of implicit and explicit control flow in instructions. In *ICML*, 2021.

[19] S. I. Wang, P. Liang, and C. D. Manning. Learning language games through interaction. *ArXiv*, abs/1606.02447, 2016.

[20] J. Oh, S. Singh, H. Lee, and P. Kohli. Zero-shot task generalization with multi-task deep reinforcement learning. *ArXiv*, abs/1706.05064, 2017.

[21] M. Jänner, K. Narasimhan, and R. Barzilay. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics*, 6:49–61, 2018.

[22] D. Bahdanau, F. Hill, J. Leike, E. Hughes, S. Hosseini, P. Kohli, and E. Grefenstette. Learning to understand goal specifications by modelling reward. In *ICLR*, 2019.

[23] K. Chen, J. Chen, J. Chuang, M. V'azquez, and S. Savarese. Topological planning with transformers for vision-and-language navigation. *ArXiv*, abs/2012.05292, 2020.

[24] K. Hermann, F. Hill, S. Green, F. Wang, R. Faulkner, H. Soyer, D. Szepesvari, W. Czarnecki, M. Jaderberg, D. Teplyashin, M. Wainwright, C. Apps, D. Hassabis, and P. Blunsom. Grounded language learning in a simulated 3d world. *ArXiv*, abs/1706.06551, 2017.

[25] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, M. Savva, S. Song, A. Zeng, and Y. Zhang. Matterport3d: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017.

[26] 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 *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[27] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In *AAAI*, 2018.

[28] H. Chen, A. Suhr, D. K. Misra, N. Snavely, and Y. Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12530–12539, 2019.

[29] J. Krantz, E. Wijmans, A. Majumdar, D. Batra, and S. Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. 2020.

[30] 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 *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. URL `https://arxiv.org/abs/1912.01734`.

[31] G. Kuhlmann, P. Stone, R. J. Mooney, and Shavlik. Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer. In *AAAI 2004*, 2004.

[32] D. L. Chen and R. J. Mooney. Learning to interpret natural language navigation instructions from observations. pages 859–865, August 2011. URL `http://nn.cs.utexas.edu/?chen:aaai11`.

[33] Y. Artzi and L. Zettlemoyer. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1:49–62, 2013. doi:10.1162/tacl_a_00209. URL `https://www.aclweb.org/anthology/Q13-1005`.

[34] J. Andreas and D. Klein. Alignment-based compositional semantics for instruction following. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1165–1174, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi:10.18653/v1/D15-1138. URL `https://www.aclweb.org/anthology/D15-1138`.

[35] T. Kollar, S. Tellex, D. Roy, and N. Roy. Grounding verbs of motion in natural language commands to robots. *Springer Tracts in Advanced Robotics*, 79, 01 2010. doi:10.1007/978-3-642-28572-1_3.

[36] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. volume 2, 01 2011.

[37] H. Mei, M. Bansal, and M. R. Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*, 2016.

[38] D. K. Misra, J. Langford, and Y. Artzi. Mapping instructions and visual observations to actions with reinforcement learning. In *EMNLP*, 2017.

[39] V. Blukis, Y. Terme, E. Niklasson, R. A. Knepper, and Y. Artzi. Learning to map natural language instructions to physical quadcopter control using simulated flight. In *CoRL*, 2019.

[40] A. Akakzia, C. Colas, P.-Y. Oudeyer, M. Chetouani, and O. Sigaud. Grounding Language to Autonomously-Acquired Skills via Goal Generation. In *ICLR 2021 - Ninth International Conference on Learning Representation*, Vienna / Virtual, Austria, May 2021. URL https://hal.inria.fr/hal-03121146.

[41] Y. Jiang, S. Gu, K. Murphy, and C. Finn. Language as an abstraction for hierarchical deep reinforcement learning. In *NeurIPS*, 2019.

[42] H. Chan, Y. Wu, J. Kiros, S. Fidler, and J. Ba. Actrce: Augmenting experience via teacher's advice for multi-goal reinforcement learning. *ArXiv*, abs/1902.04546, 2019.

[43] J. D. Co-Reyes, A. Gupta, S. Sanjeev, N. Altieri, J. DeNero, P. Abbeel, and S. Levine. Guiding policies with language via meta-learning. *ArXiv*, abs/1811.07882, 2019.

[44] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. Nguyen, and Y. Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. In *ICLR*, 2019.

[45] J. D. Kanu, E. Dessalene, X. Lin, C. Fermuller, and Y. Aloimonos. Following instructions by imagining and reaching visual goals. *ArXiv*, abs/2001.09373, 2020.

[46] R. Kaplan, C. Sauer, and A. Sosa. Beating atari with natural language guided reinforcement learning. *ArXiv*, abs/1704.05539, 2017.

[47] P. Goyal, S. Niekum, and R. Mooney. Using natural language for reward shaping in reinforcement learning. *ArXiv*, abs/1903.02020, 2019.

[48] X. E. Wang, Q. Huang, A. Çelikyilmaz, J. Gao, D. Shen, Y. Wang, W. Y. Wang, and L. Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6622–6631, 2019.

[49] S. P. Mirchandani, S. Karamcheti, and D. Sadigh. Ella: Exploration through learned language abstraction. *ArXiv*, abs/2103.05825, 2021.

[50] P. Goyal, S. Niekum, and R. Mooney. Pixl2r: Guiding reinforcement learning using natural language by mapping pixels to rewards. *ArXiv*, abs/2007.15543, 2020.

[51] G. Cideron, M. Seurin, F. Strub, and O. Pietquin. Self-educated language agent with hindsight experience replay for instruction following. *ArXiv*, abs/1910.09451, 2019.

[52] S. Dasari, F. Ebert, S. Tian, S. Nair, B. Bucher, K. Schmeckpeper, S. Singh, S. Levine, and C. Finn. Robonet: Large-scale multi-robot learning. In *Conference on Robot Learning*, 2019.

[53] A. S. Chen, H. Nam, S. Nair, and C. Finn. Batch exploration with examples for scalable robotic reinforcement learning. *IEEE Robotics and Automation Letters*, 2021.

[54] 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 *Conference on Robot Learning*, 2020.

[55] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *ArXiv*, abs/2104.08212, 2021.

[56] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning. In *CoRL*, 2017.

[57] T. Yu, C. Finn, S. Dasari, A. Xie, T. Zhang, P. Abbeel, and S. Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

[58] A. S. Chen, S. Nair, and C. Finn. Learning generalizable robotic reward functions from ”in-the-wild” human videos. *ArXiv*, abs/2103.16817, 2021.

[59] T. Zhao, A. Nagabandi, K. Rakelly, C. Finn, and S. Levine. Meld: Meta-reinforcement learning from images via latent state models. *ArXiv*, abs/2010.13957, 2020.

[60] L. P. Kaelbling. Learning to achieve goals. In *IN PROC. OF IJCAI-93*, pages 1094–1098. Morgan Kaufmann, 1993.

[61] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. H. Fong, P. Welinder, B. McGrew, J. Tobin, P. Abbeel, and W. Zaremba. Hindsight experience replay. *ArXiv*, abs/1707.01495, 2017.

[62] S. Nasiriany, V. H. Pong, A. Nair, A. Khazatsky, G. Berseth, and S. Levine. Disco rl: Distribution-conditioned reinforcement learning for general-purpose policies. *ArXiv*, abs/2104.11707, 2021.

[63] B. Eysenbach, R. Salakhutdinov, and S. Levine. Search on the replay buffer: Bridging planning and reinforcement learning. In *NeurIPS*, 2019.

[64] Y. Chebotar, K. Hausman, Y. Lu, T. Xiao, D. Kalashnikov, J. Varley, A. Irpan, B. Eysenbach, R. C. Julian, C. Finn, and S. Levine. Actionable models: Unsupervised offline reinforcement learning of robotic skills. *ArXiv*, abs/2104.07749, 2021.

[65] F. Ebert, C. Finn, S. Dasari, A. Xie, A. Lee, and S. Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv:1812.00568*, 2018.

[66] Y.-C. Lin, M. Bauzá, and P. Isola. Experience-embedded visual foresight. *ArXiv*, abs/1911.05071, 2019.

[67] S. Tian, S. Nair, F. Ebert, S. Dasari, B. Eysenbach, C. Finn, and S. Levine. Model-based visual planning with self-supervised functional distances. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=UcoXdfrORC`.

[68] J. Fu, A. Singh, D. Ghosh, L. Yang, and S. Levine. Variational inverse control with events: A general framework for data-driven reward definition. In *Advances in Neural Information Processing Systems*, 2018.

[69] A. Singh, L. Yang, C. Finn, and S. Levine. End-to-end robotic reinforcement learning without reward engineering. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019.

[70] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *ArXiv*, abs/1910.01108, 2019.

[71] M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine. Stochastic variational video prediction. *arXiv:1710.11252*, 2017.

[72] B.-H. Wu, S. Nair, R. Martín-Martín, L. Fei-Fei, and C. Finn. Greedy hierarchical variational autoencoders for large-scale video prediction. *ArXiv*, abs/2103.04174, 2021.

[73] R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[74] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018.

[75] S. Nair and C. Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. *ArXiv*, abs/1909.05829, 2020.

[76] Anonymous. Example-driven model-based reinforcement learning for solving long-horizon visuomotor tasks.

A Method Implementation Details

A.1 LORL

We begin by describing the method implementation details for our method LORL.

A.1.1 Reward Function

The reward function is trained as a binary classifier which takes as input the initial state s_0, the current state s, and language instruction l, and outputs a scalar [0,1] prediction.

Architecture. The network first concatenates the initial image (64,64,3) and goal image (64,64,3) channel wise. The resulting image is passed through a convolutional image encoder with ReLU activations with the following architecture [channels, kernel size, stride] (all with padding 1): [[32, 4, 2], [32, 4, 1], [64, 4, 2], [64, 4, 2], [128, 4, 2], [128, 4, 1], [256, 4, 2], [256, 4, 1]]. The output is flattened and passed through fully connected layers of size [512, 512, 512, 512, L] where L is a hyperparameter for the image embedding size. In our real robot experiments the observations have 4 camera views, each (64, 64, 3) so we concatenate each initial image and goal per view, then use the same convolutional architecture but with 4 groups.

Simultaneously, the language instruction is passed through a pretrained `distilbert-base-uncased sentence` encoder [70], available at `https://huggingface.co/distilbert-base-uncased`. The network performs sub-word tokenization using the distilber tokenizer, then passes the sentences through the 15 layer transformer network, outputting a real valued vector of size 768.

The resulting image encoding of size L and the sentence encoding of size 768 are concatenated, and fed through a fully connected network of size $[L, L, L, 1]$ where the first three layers have ReLU activations and a 0.2 dropout, and the final scalar prediction goes through a Sigmoid activation.

Hyper-parameters. In our experiments we train the reward with $L = 128$, batchsize of 32, and Adam optimizer with learning rate 0.00001. In our simulated experiments we use $\alpha = 0$, that is, we only use the initial and final states of the episode for positives. In our real robot experiments we use $\alpha = 0.25$, selecting positives with initial state in the first quarter of the episode and final states from the last quarter of the episode.

Augmentation. To prevent the classifier from over-fitting, we use visual data augmentation in the form of color jitter (brightness=0.02, contrast=0.02, saturation=0.02, hue=0.02) and affine transformations of up to 20 pixels on the images (translate=(0.1, 0.1), scale=(0.9, 1.1)). We also add uniform random noise in $[-0.1, 0.1]^{768}$ to the instruction embeddings.

A.1.2 Visual Dynamics Model (Sim)

In our simulation experiments, we train an action-conditioned video prediction model using Stochastic Variational Video Prediction [71]. We train it on the full simulated dataset of 1M frames for 300000 iterations with all default hyperparameters using the codebase `https://github.com/tensorflow/tensor2tensor`.

A.1.3 Visual Dynamics Model (Real Robot)

On the real robot setup we leverage a pre-trained visual dynamics model on the robot desk setup using the GHVAE [72] architecture, that trains a VAE, encodes each of the 4 camera views, and learns a predictive model in the latent space.

A.1.4 MPC Planner

Finally for choosing actions with the learned reward and visual dynamics model we use model predictive control. Specifically, in simulation we optimize a single sequence of 20 actions which are stepped in the environment. We sample 200 action sequences of length 20, which are fed through the video prediction model and ranked according to the reward of their final state. The top 10% are used to refit the action sampling distribution, and the process is repeated 3 times before the best action sequence is stepped in the environment.

On the real robot, we optimize action sequences of length 5, using again 3 iterations of CEM, now with 48 sampled action sequences per iteration. After stepping each 5 step action sequence we step the agent in the environment, then repeat for the full 30 timestep episode. On the real robot, we use the same initial state T=0 as the initial state throughout the full episode.

A.2 Language Conditioned Imitation Learning

The language conditioned imitation learning agent takes in the current state s and language instruction l from that episode and is trained to minimize the mean squared error to the action taken in the data.

Architecture. This agent uses an identical image encoder, and identical distilbert sentence encoder as LORL. The resulting embeddings are again concatenated and fed through a fully connected network of size $[L, L, L, A]$ where the first three layers have ReLU activations and a 0.2 dropout, and the final prediction outputs the A dimensional action. **Hyper-parameters.** This agent is trained with $L = 128$, batch-size of 32 and an Adam optimizer with learning rate 0.0001. **Augmentation.** The BC agent uses the same visual augmentation as LORL. Unlike LORL which struggles with over-fitting to language instructions, the BC agent struggles with under-fitting and learning different behavior for different instructions, hence we omit the uniform noise in language embedding space.

A.3 Language Conditioned Reinforcement Learning

The language-conditioned Q learning agent learns a language conditioned Q function which takes as input the initial state s_0, current state s, action a, and language instruction l and aims to predict the discounted return for taking action a in state s. It is trained on balanced batches of positive transitions which are terminal states with reward 1, where the current state is the final state for the episode which was labeled with instruction l, and negative transitions where the current state is any state earlier in the episode and has reward 0. The Q function is then trained with the standard Bellman error, where target Q values which maximize s_{t+1} are compute by sampling M actions and taking the one with the highest Q value. During evaluation at each timestep the agent selects the action which maximizes the Q value at the current state via sampling M actions and taking the best one.

Architecture. The Q function uses an identical architecture to the LORL classifier, with the exception that it also takes as input the action a which is concatenated at each of the final 3 fully connected layers. **Hyper-parameters.** We use $L = 128$, Adam optimizer with learning rate 0.0001, $M = 100$, and batch size 8. **Augmentation.** Like the BC agent, the Q-learning agent uses the same visual augmentation as LORL but omits noise in the language embedding space to help the policy learn to differentiate between different instructions.

A.4 Oracle, Random, Pixel, LPIPS Baseline

The Oracle performance is computed using the exact same planner as LORL, but uses a ground truth dynamics model and a ground truth cost computed on the environment state. The random policy simply samples random actions at each time steps. The Pixel baseline first receives a goal image in which the object has been moved to its desired goal position (which constitutes task success). It then uses the exact same visual dynamics model and planner, but with a cost function which minimizes the ℓ_2^2 pixel distance between the goal image and the final image in each trajectory. The LPIPS baseline uses the same planner, dynamics, and goal image, but uses LPIPS similarity instead of negative pixel distance.

B Environment/Data Details

B.1 Simulation Environment

The simulated environment is built off of Meta-World [54], and consists of a simulated sawyer arm interacting over a table with a drawer, a faucet, and two mugs. The agent is controlled using delta end effector control, and the episode horizon in the environment is 20 timesteps. The observation space is (64, 64, 3) RGB images.

B.2 Simulation Data and Annotation

We collect 50000 episodes in the simulated environment (each 20 timesteps) for a total of 1M frames. Data is collected by a random policy which samples uniformly in the action space. During data collection the arm and scene is reset randomly each episode. To annotate each episode, we procedurally look at the true environment state at the beginning of the episode and at the end of the episode and record the change in position of the objects (including the drawer, faucet, and mugs). For each object change in state we generate a phrase based on the measured change, such as “move the black mug left” or “turn the faucet right”. Then all the phrases for a single episode and shuffled and combined with an “and” to create the final annotation for that episode. If no object moves the episode is annotated with “Do nothing”. Doing so produces around 2300 unique annotations total.

B.3 Robot Environment

The robot environment consists of a real Franka Emika Panda robot operating over an Ikea desk which consists of a cabinet, 2 drawers, and numerous objects. The robot has 4 camera views, one one which is a side view of the table, one which is a birds eye view from the right side of the table, one which is a birds eye view from the left side of the table, and one wrist mounted camera. Each returns (64,64,3) RGB images. The robots action space includes delta end-effector control (up to 7cm) and grasping (however we do not use the grasping capability in any of our evaluation tasks).

B.4 Robot Data and Annotation

Our robot dataset consists of 3000 episodes, each 50 timesteps, taken directly without modification from concurrent work [76] which aims to learn a diverse det of skill using online RL. As a result, this data is from an autonomously trained RL agent, and contains a mix of meaningful and close to random behavior. Moreover, even when tasks are completed they may be completed in highly sub-optimal ways. Some of the tasks the agent is trained for include opening the drawers and cabinets, grasping and placing different objects, and inserting markers/plugs.

We then have each episode annotated using crowdsourcing, specifically Amazon Mechanical Turk (See Figure 8). We collect 2 annotations per episode for a total of 6000 annotations. During training, we filter out episodes where the annotators said the robot did nothing or they could not tell what the robot was doing, resulting in a final dataset of 1600 episodes. We visualize the top 25 annotations and their counts in the filtered dataset in Figure 9.

C Experimental Evaluation/Task Details

Next we describe the experimental setup in detail, specifically the tasks and their evaluation criteria.

C.1 Simulation Tasks.

The simulation tasks used for evaluation are (1) closing the drawer, (2) opening the drawer, (3) turning the faucet left, (4) turning the faucet right, (5) moving the black mug right, and (6) moving the white mug down. For language conditioned approaches these tasks are specified with “close drawer”, “open drawer”, “turn faucet left”, “turn faucet right”, “move black mug right”, and “move white mug down” respectively. For goal image comparisons, we generate goal images with the obejct moved to a position which satisfies the task, see Figure 10 for examples. For all drawer and mug tasks success is if the object moves in the correct direction by at least 2cm at any point in the episode. For the faucet it is if it rotates at least $\pi/10$ radians in the right direction at any point in the episode.

C.2 Experiment 1 Details

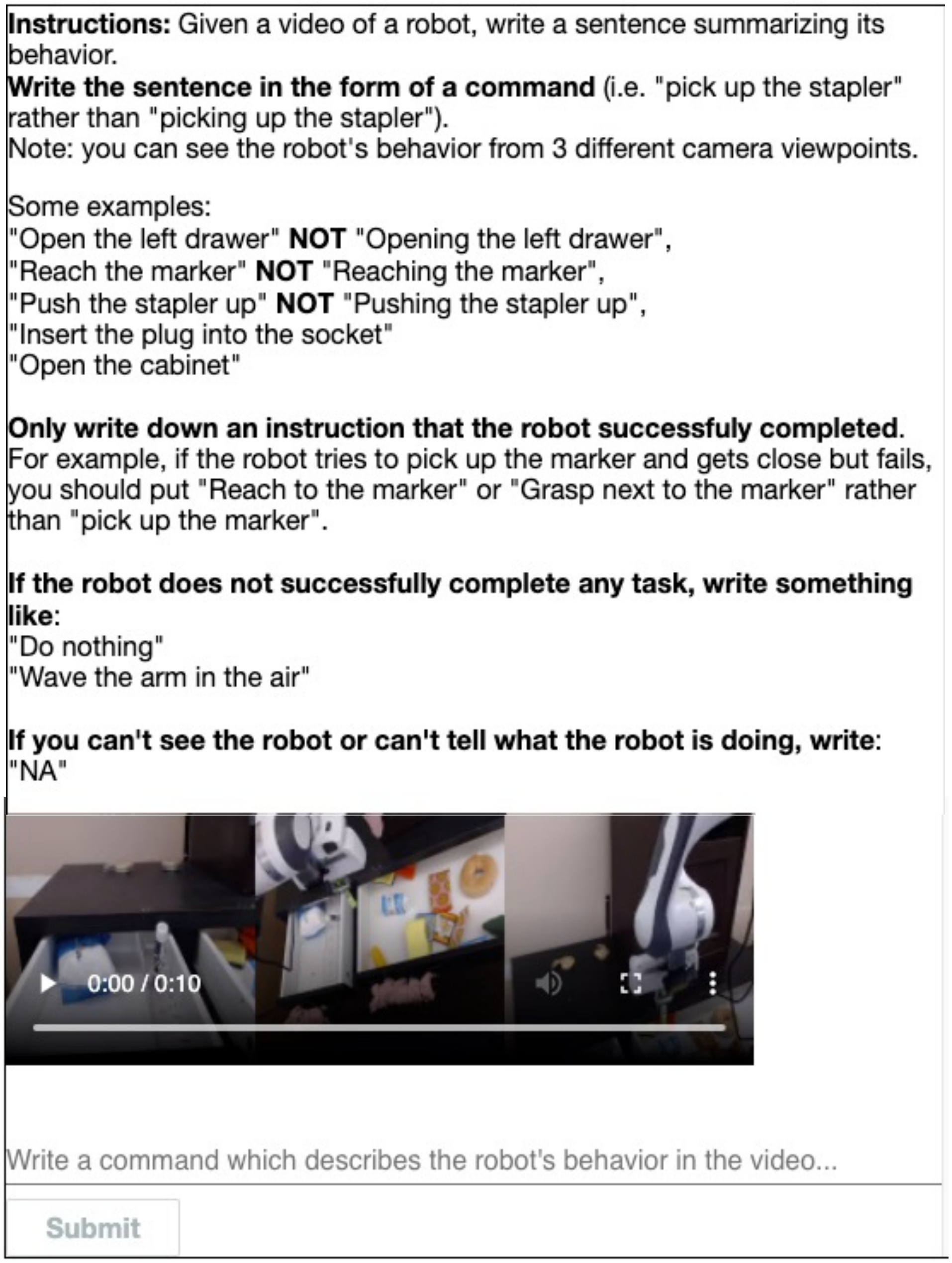
For experiment 1 we compare each of the methods on the 6 tasks described previously. Using the same 50000 episode dataset we train 3 seed each of our method, the imitation baseline, and the RL baseline. Our method and the imitation baseline are trained for 390000 iterations, while the RL agent is trained until the Bellman loss plateaus at around 240000 iterations. For each task and each seed we conduct 100 random trials (with random initialization) to compute a success rate out of 100. For the oracle, pixel, and random we directly run 300 random trials per task.

C.3 Experiment 2 Details

In experiment 2 we consider the same setup as experiment 1, but rephrase the instructions as described in the main text. The full set of rephrased instructions can be found in Figure 11. For the human provided unseen natural language, we sample one of the provided instructions at random for each episode.

C.4 Robot Tasks

On the real robot, we consider 5 tasks, (1) opening the left drawer, (2) opening the right drawer, (3) moving a stapler, (4) reach the markers in the left drawer, and (5) reaching the cabinet.



Instructions: Given a video of a robot, write a sentence summarizing its behavior.
Write the sentence in the form of a command (i.e. "pick up the stapler" rather than "picking up the stapler").
Note: you can see the robot's behavior from 3 different camera viewpoints.

Some examples:
"Open the left drawer" **NOT** "Opening the left drawer",
"Reach the marker" **NOT** "Reaching the marker",
"Push the stapler up" **NOT** "Pushing the stapler up",
"Insert the plug into the socket"
"Open the cabinet"

Only write down an instruction that the robot successfuly completed.
For example, if the robot tries to pick up the marker and gets close but fails, you should put "Reach to the marker" or "Grasp next to the marker" rather than "pick up the marker".

If the robot does not successfully complete any task, write something like:
"Do nothing"
"Wave the arm in the air"

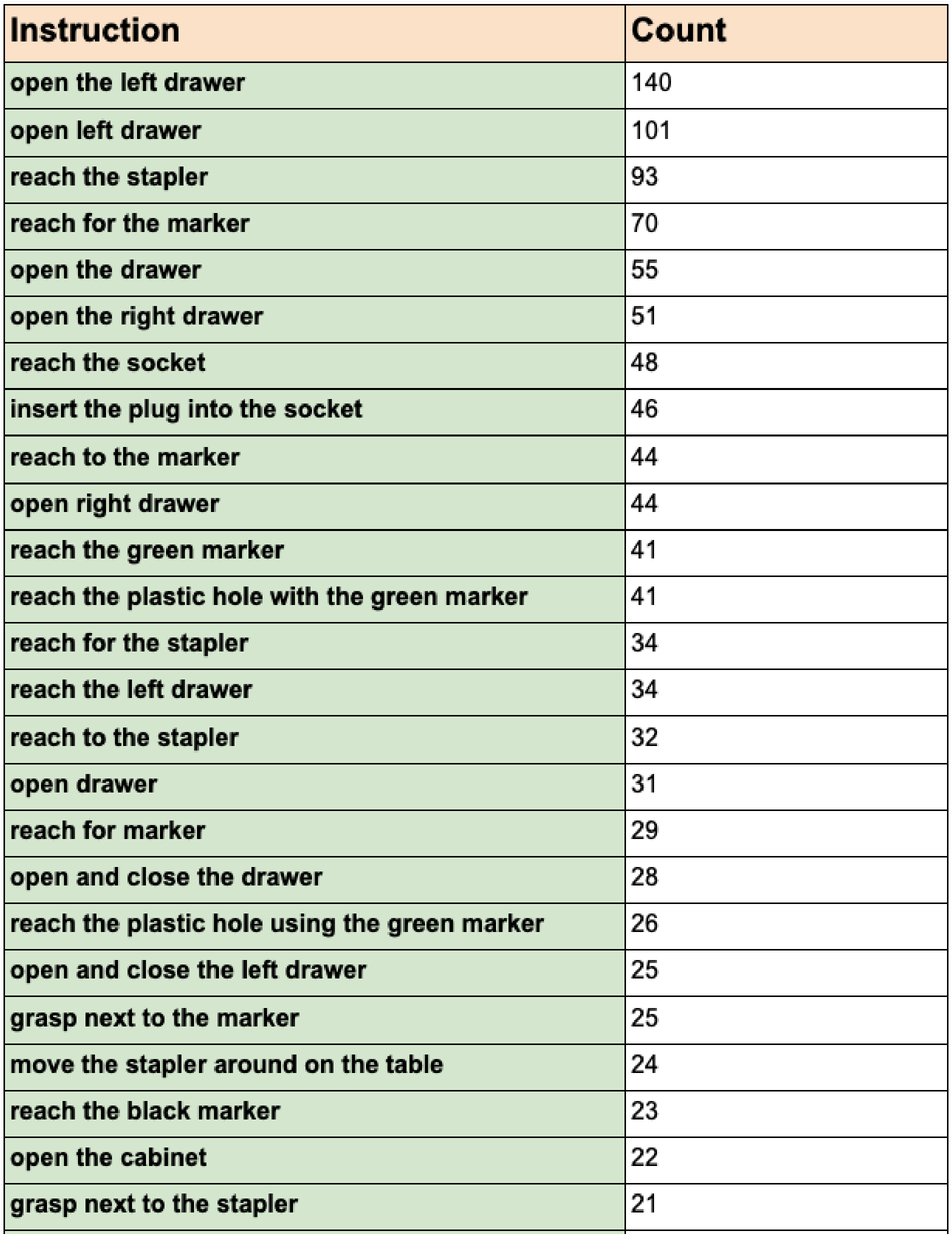
If you can't see the robot or can't tell what the robot is doing, write:
"NA"

Write a command which describes the robot's behavior in the video...

Figure 8: **Annotation interface.** The annotators are shown a video of the episode, and are given the above instructions. We collect 6000 annotations, 2 for each of the 3000 episodes of data.

The success criteria for task (1) and (2) is to open the drawer at least 1 inch from its initial position at any point in the episode. For task (3) it is to translate the stapler on the table by any amount. For task (4) it is to reach the gripper tip within 1 inch of any marker in the left drawer. For task (5) it is making contact with the cabinet.

The initial state for task (1) is the robot above and to the right of the left drawer, and the drawer starts slightly open to make grasping easier. The initial state for task (2) is above and to the left of the right drawer, and again the drawer starts slightly open. For task (3) the robot starts near the middle/right side of the desk, and the stapler is initialized randomly on the front half of the desk. For task (4) and (5) the robot has the same initialization as in task (1), and in task (4) the markers are randomly moved around within the left drawer. For each episode of each task the initial position is randomized up to 5cm in any direction from the main initial position. All resets and successes are measured by



Instruction	Count
open the left drawer	140
open left drawer	101
reach the stapler	93
reach for the marker	70
open the drawer	55
open the right drawer	51
reach the socket	48
insert the plug into the socket	46
reach to the marker	44
open right drawer	44
reach the green marker	41
reach the plastic hole with the green marker	41
reach for the stapler	34
reach the left drawer	34
reach to the stapler	32
open drawer	31
reach for marker	29
open and close the drawer	28
reach the plastic hole using the green marker	26
open and close the left drawer	25
grasp next to the marker	25
move the stapler around on the table	24
reach the black marker	23
open the cabinet	22
grasp next to the stapler	21

Figure 9: **Top Instructions.** We list the top 25 instructions and the number of times they appear in the data in the filtered robot dataset.

human supervisor. See the website `https://sites.google.com/view/robotlorl` for videos of task completion.

C.5 Experiment 3 Details

In the real robot experiment we conduct 10 trials of each task for our method and for an ablation of our method trained without flipped initial/final negatives. The episode horizon for each task is 30 timesteps, and the agent plans 5 actions at a time as described in Section A.1.4. We also run 10 trials of tasks (1) and (3) with more sophisticated instructions.

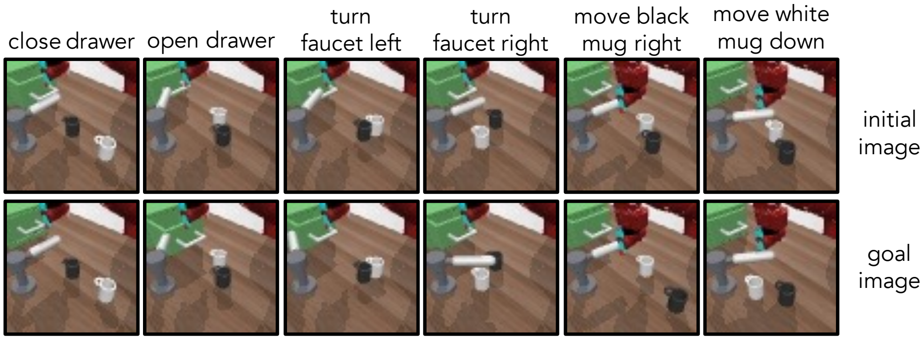


Figure 10: **Example Initial/Goal Images.** Examples of generated goal images for the Pixel baseline.

D Additional Results

D.1 Additional Ablations

First, we run an additional ablation in simulation which ablates the effect of both types of negative examples. When removing the randomly chosen negatives from different episodes, we see a dramatic drop in performance (average success rate 56% to 27%), as this is the main way the agent learns the effect of different language instructions. When removing the flipped negatives in simulation, we see limited impact (average success rate 56% to 55%). This is because the primary role of the flipped negatives are to prevent over-fitting to objects in the scene, and to capture temporal progress, and in simulation the scene is fixed, and has a large amount (50K) episodes of training data. This is in contrast to the real robot, with 1.6K episodes, and many different scenes, and in that setting removing the flipped negatives drops average success rate from 66% to 36%.

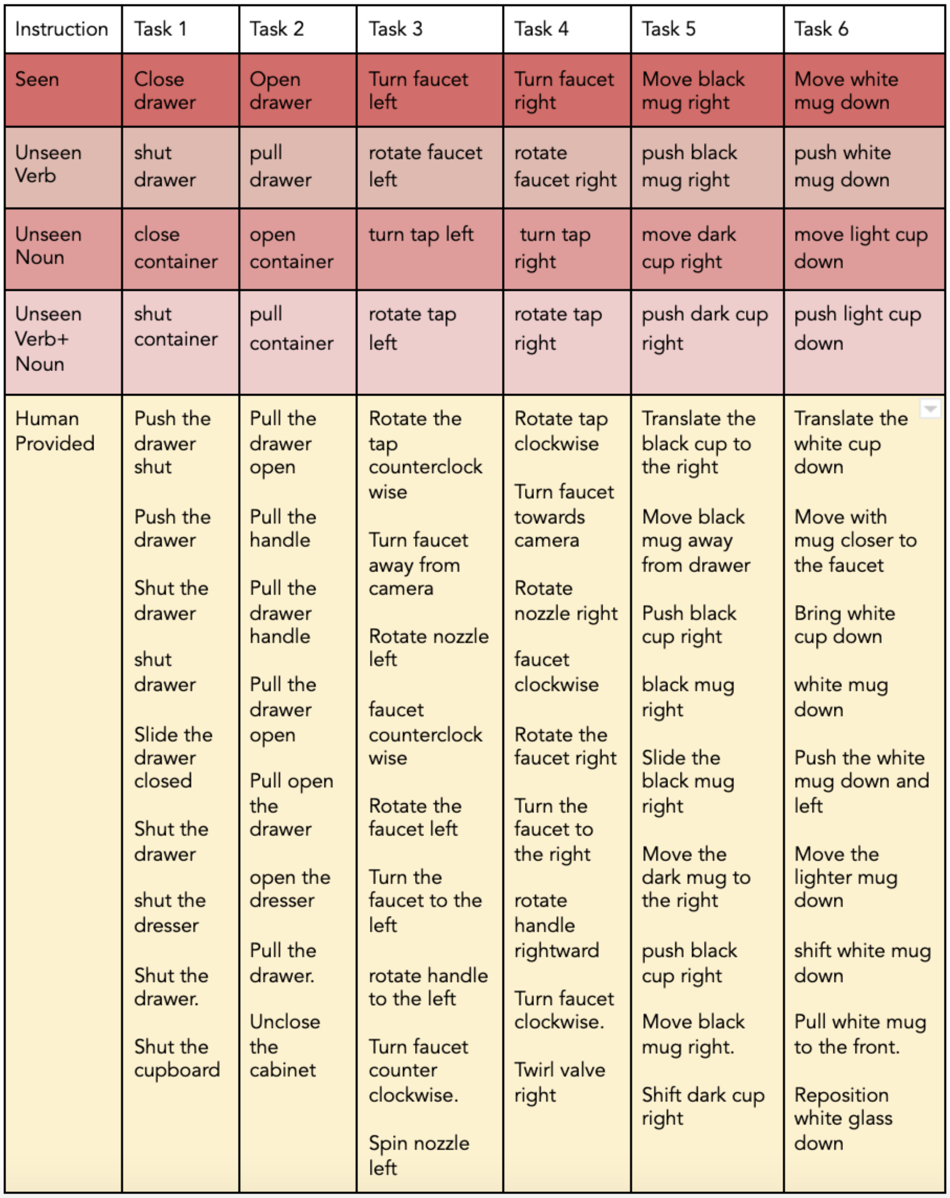
Task (10 Trials Each)	LORL	LORL (-Filtering)	LORL (-NP)
"Open the left drawer"	**90**%	70%	30%
"Open the right drawer"	**40**%	20%	30%
"Move the stapler"	50%	10%	**70**%
'Reach the marker"	**70**%	0%	10%
"Reach the cabinet"	**80**%	30%	20%
Average over tasks	**66**%	26%	32%

Table 3: **Additional Real Robot Ablations.** Our ablations suggest that both filtering out episodes which annotators labeled as "do nothing" and training with noisy positives are important for real robot performance.

Second, In Table 3, we run two more ablations on the real robot, where we consider (1) removing the data cleaning/filtering step where we remove episodes where the annotators indicated the robot did nothing **(LORL (- Filtering))** and (2) where we do not include noisy positives to generate more positive examples during training **(LORL (- NP))**. First, we find removing filtering reduces performance considerably. In many cases annotators could not tell what the robot was doing, or wrote "do nothing" when they were unsure, making these labels particularly noisy. As a result, we filtered out these episodes primarily as a way of cleaning out noise in the data. While our simulation results suggest that our method can handle some behavior that completes no task ($\sim 20\%$ of the dataset), in the extreme case where almost half the data is labeled as not doing a task (as is the case in the real robot data), it can make learning an effective reward more difficult. Specifically, episodes of "do nothing" may dominate training batches and negative examples, making it harder to learn the difference between different language instructions, especially instructions that are infrequent in the data. In practice, post-hoc cleaning/filtering of the data requires little to no extra supervision, and can make training the reward considerably easier. Second, we find that generating more positive examples via noisy labeling is important in the robot domain, where there are only 1.6K episodes of data (after filtering/cleaning).

D.2 LORL Training Curves

In Figure 12 (left) we include the train/test accuracy curves for LORL in simulation, as well as the ablations of LORL in simulation (LORL, LORL without the randomly chosen negatives, and LORL without the flipped negatives). As expected, removing random negatives makes fitting the data much easier, but leads to a worse reward and worse planning performance. In Figure 12 (right) we include the train/test accuracy curves for LORL on real robot data, including LORL, LORL without filtering, LORL without flipped negatives, and LORL without noisy positives). We see that without filtering,



Instruction	Task 1	Task 2	Task 3	Task 4	Task 5	Task 6
Seen	Close drawer	Open drawer	Turn faucet left	Turn faucet right	Move black mug right	Move white mug down
Unseen Verb	shut drawer	pull drawer	rotate faucet left	rotate faucet right	push black mug right	push white mug down
Unseen Noun	close container	open container	turn tap left	turn tap right	move dark cup right	move light cup down
Unseen Verb+ Noun	shut container	pull container	rotate tap left	rotate tap right	push dark cup right	push light cup down
Human Provided	Push the drawer shut Push the drawer Shut the drawer shut drawer Slide the drawer closed Shut the drawer shut the dresser Shut the drawer. Shut the cupboard	Pull the drawer open Pull the handle Pull the drawer handle Pull the drawer open Pull open the drawer open the dresser Pull the drawer. Unclose the cabinet	Rotate the tap counterclockwise Turn faucet away from camera Rotate nozzle left faucet counterclockwise Rotate the faucet left Turn the faucet to the left rotate handle to the left Turn faucet counter clockwise. Spin nozzle left	Rotate tap clockwise Turn faucet towards camera Rotate nozzle right faucet clockwise Rotate the faucet right Turn the faucet to the right rotate handle rightward Turn faucet clockwise. Twirl valve right	Translate the black cup to the right Move black mug away from drawer Push black cup right black mug right Slide the black mug right Move the dark mug to the right push black cup right Move black mug right. Shift dark cup right	Translate the white cup down Move with mug closer to the faucet Bring white cup down white mug down Push the white mug down and left Move the lighter mug down shift white mug down Pull white mug to the front. Reposition white glass down

Figure 11: **Example Instructions used in Experiment 2.**

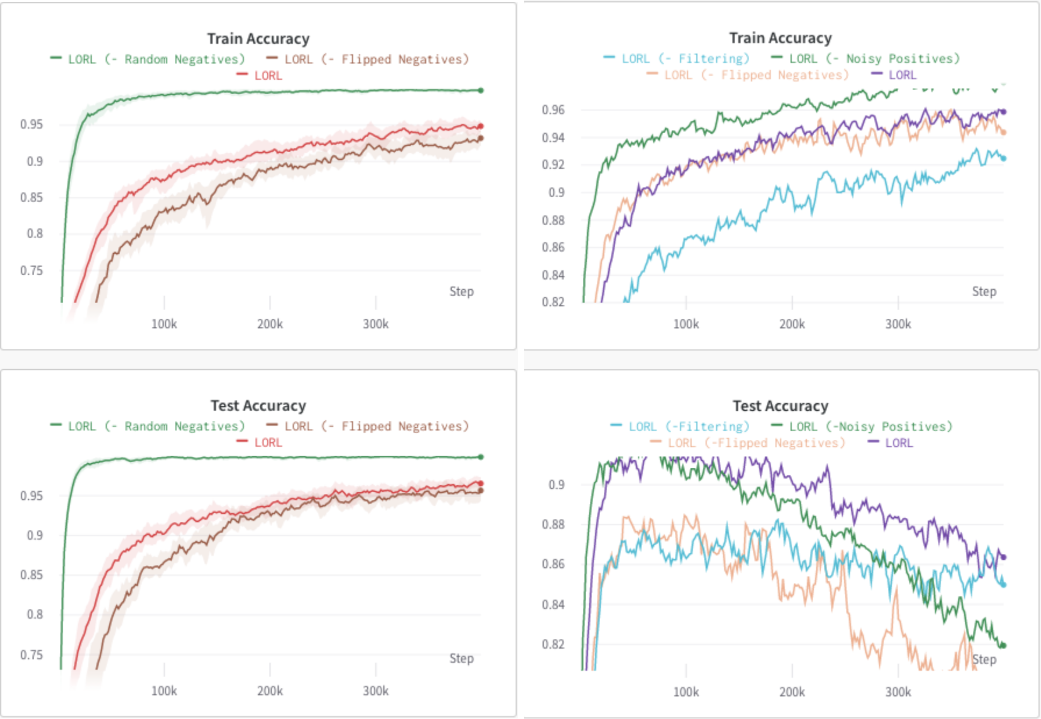


Figure 12: **LORL Learning Curves** Learning Curves for LORL and ablations in simulation (left) and on the real robot data (right).

fitting the data is much more difficult, and without the noisy positives, fitting the data is easier (but can also leads to a worse reward function for planning). For all ablations the last checkpoint before test accuracy starts decreasing is used in planning.

D.3 Unseen Instruction Generalization Results

In Figure 13 we include the per task success rates for our method when using the original command vs. unseen variations.

D.4 Qualitative Examples

We include qualitative examples of the ranked predicted trajectories under different language instructions on the real robot in Figures 14, 15, 16, 17, and 18.

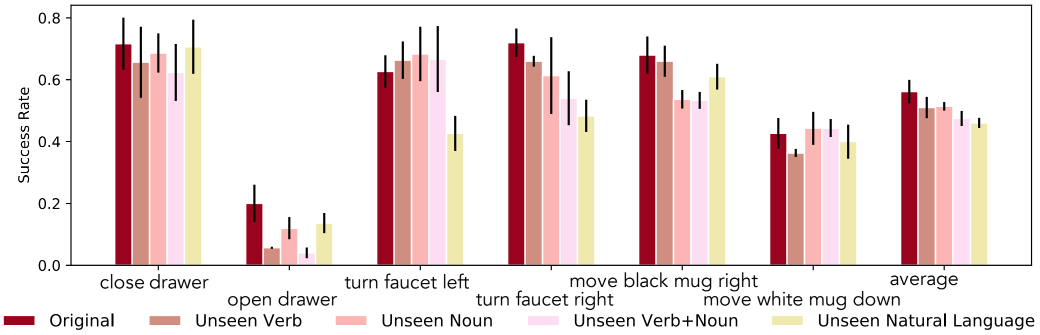


Figure 13: **Generalization to Unseen Commands**. We observe that by leveraging pre-trained language models, LORL is able to generalize to unseen instructions with unseen verbs, nouns, and human generated instructions with a minimal ($\leq 10\%$) drop in performance. Success rates computed over 3 seeds of 100 trials.

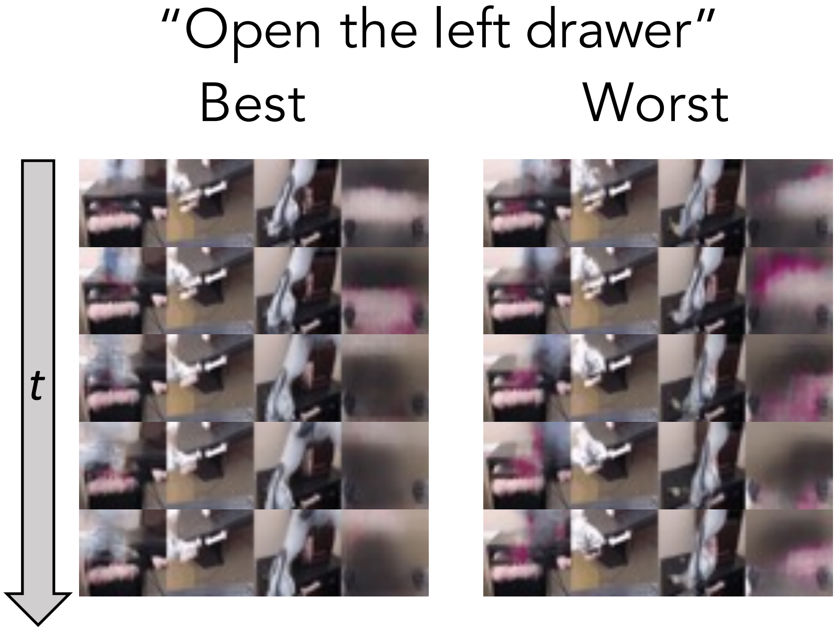


Figure 14: **LORL Ranking.** Top and bottom ranked trajectories for "Open the left drawer".

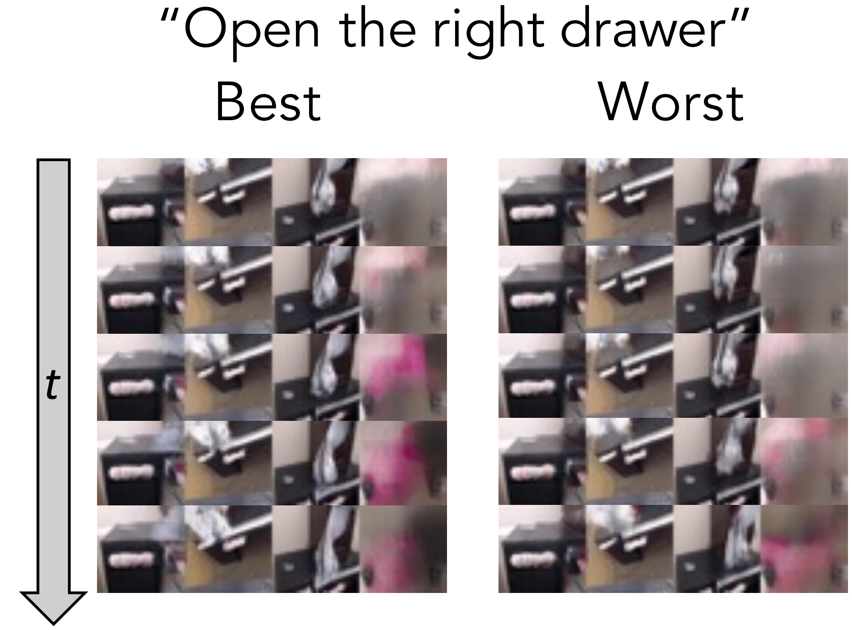


Figure 15: **LORL Ranking.** Top and bottom ranked trajectories for "Open the right drawer".

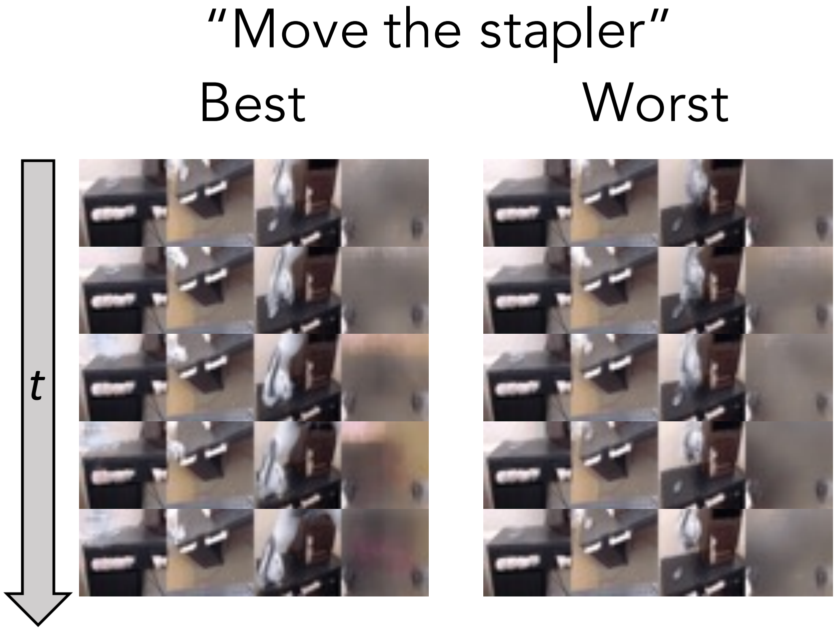


Figure 16: **LORL Ranking.** Top and bottom ranked trajectories for "Move the stapler".

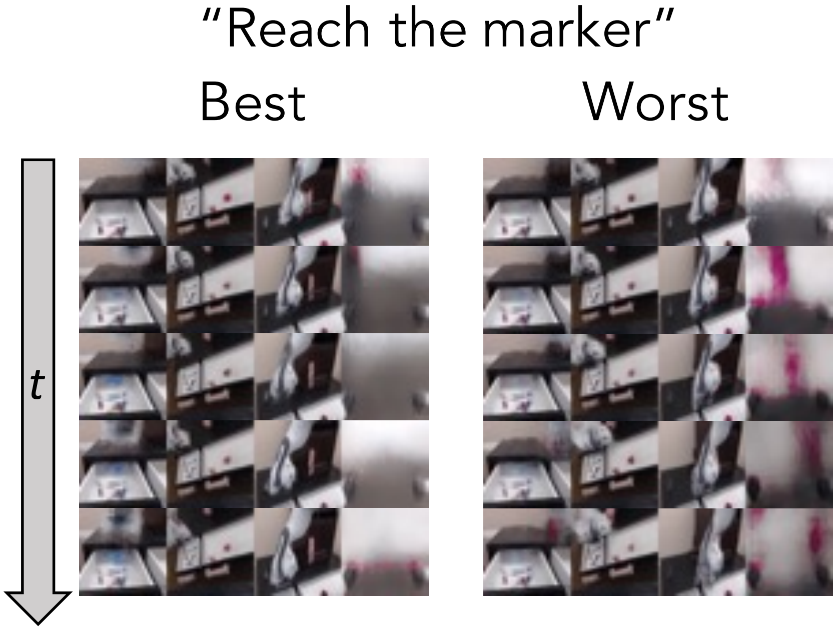


Figure 17: **LORL Ranking.** Top and bottom ranked trajectories for "Reach the marker".

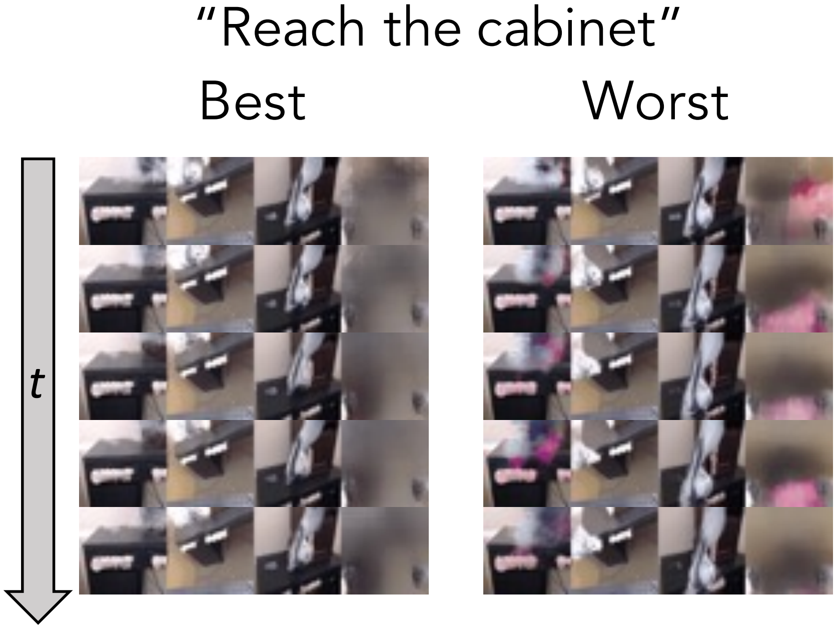


Figure 18: **LORL Ranking.** Top and bottom ranked trajectories for "Reach the cabinet".