Policy Learning with a Language Bottleneck

Anonymous authors
Paper under double-blind review

Abstract

Modern AI systems such as self-driving cars and game-playing agents achieve superhuman performance. But they often lack human-like generalization, interpretability, and interoperability with human users. This paper introduces *Policy Learning with a Language Bottleneck* (PLLB), a framework enabling AI agents to generate linguistic rules that capture the high-level strategies underlying rewarding behaviors. PLLB alternates between a *rule generation* step guided by language models, and an *update* step where agents learn new policies guided by rules. Crucially, PLLB enables this kind of language-guided learning even when a natural language rule is insufficient to completely describe the target policy. Across five diverse tasks, including a two-player signaling game, maze navigation, image reconstruction, and robot grasp planning, we show that PLLB learns more interpretable and generalizable behaviors than standard policy learning methods. In three additional human subject studies, we show that show the learned rules significantly improve *human* task performance, enabling more effective human-AI coordination.[1]

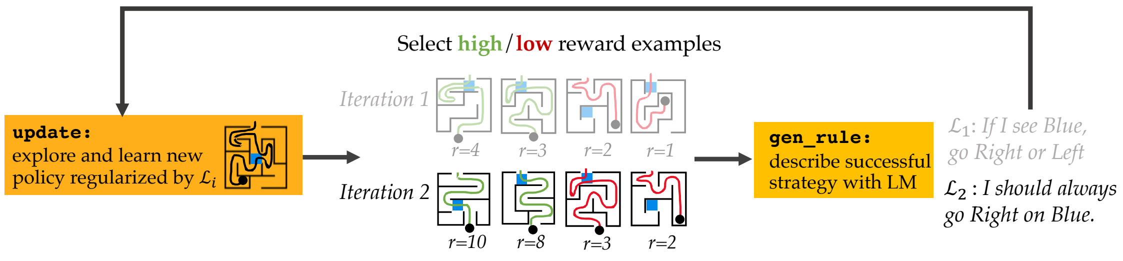

Select high/low reward examples
update:
explore and learn new policy regularized by $\mathcal{L}_i$
Iteration 1
r=4
r=3
r=2
r=1
Iteration 2
r=10
r=8
r=3
r=2
gen_rule:
describe successful strategy with LM
$\mathcal{L}_1$: If I see Blue, go Right or Left
$\mathcal{L}_2$: I should always go Right on Blue.

Figure 1: *Policy Learning with a Language Bottleneck* (PLLB) alternates between two steps: 1) **`gen_rule`** generates a linguistic rule $\mathcal{L}_i$ explaining the agent's best behaviors by prompting a language model with contrastive (positive and negative) episodes; 2) **`update`** learns a new policy conditioned on $\mathcal{L}_i$. PLLB strengthens human-AI coordination by constraining policies to be more interpretable and generalize better.

1 Introduction

As AI systems play an increasingly central role in automation and decision-making, their success depends not only on their performance, but also on their ability to model and align with human behavior. To be truly effective assistants, these systems must act and generalize in ways that align with human expectations. However, many of today's AI systems do not meet these standards. Self-driving cars or game-playing agents like AlphaZero may achieve super-human performance, but they lack interpretability (McIlroy-Young et al., 2020) and often act unpredictably, especially outside their training distribution (Wang et al., 2023).

Unlike AI systems that are trained in isolation, humans acquire most of their skills and knowledge from interacting with others, often using language — via instructions, advice, or explanations that improve their decision-making capabilities (Carruthers & Boucher, 1998; Mesoudi & Thornton, 2018). Language acts as a *communicative* medium, enabling us to teach, learn from, and coordinate with others to solve complex

[1]We provide source code for our experiments at [REDACTED].

problems. It also supports other *cognitive* functions: even when not used for communication, it allows us to represent abstract concepts (Hesse, 1988; Lakoff & Johnson, 2008), and plan (Vygotsky, 1965; Clark, 1998); it guides our attention (Waxman, 1994; Yoshida & Smith, 2003), and prompts relational thinking (Gentner & Loewenstein, 2002).

Consider a driver learning to navigate novel social conventions (e.g., triangle-shaped stop signs). While adapting to the environment, they might verbalize strategies to themselves to avoid future mistakes (e.g., *If the sign is triangular, I should stop*, a cognitive use), or transmit this convention to others (e.g., telling a friend *In Japan, stop signs are triangles*, a communicative use). Representing learned information in language helps humans solve problems and transmit knowledge by effectively capturing abstract problem structures that facilitate learning and generalization (Boutonnet & Lupyan, 2015; Chopra et al., 2019; Tessler et al., 2021). Importantly, language remains useful even when it cannot fully encapsulate an entire strategy (e.g., a drivers' reflexive actions, or fine-grained driving mechanics).

There exist many recent examples in the literature of AI systems leveraging language-based representations or linguistic feedback, but these often rely on external supervision: humans in the loop or hard-coded feedback functions (Luketina et al., 2019; Colas et al., 2022). We argue that more human-like AI systems should not only *use* language-based supervision but also *generate their own language-based feedback* to leverage both the communicative and cognitive functions of language. Importantly, this capability should also extend to tasks that require updating an underlying policy that is only partially expressible in natural language, such as low-level control or reflexive actions.

This paper introduces *Policy Learning with a Language Bottleneck* (PLLB), a framework that provides artificial embodied agents the ability to generate linguistic rules that capture the strategies underlying their most rewarding behaviors. As shown in Figure 1, PLLB alternates between a **`rule generation`** step that explains the agent's experiences by prompting a language model (LM) with contrastive episodes, and a **`policy update`** step that learns a new policy guided by these rules. Unlike past work that solely leverages LMs for modeling agent behavior and multistep reasoning (Park et al., 2023; Wei et al., 2023), PLLB is applicable even when aspects of the target policy cannot be expressed with language.

Policy Learning with a Language Bottleneck can be applied to a wide range of agent types, from RL policies to LLM-based learners to robot pose estimators. We thus investigate its role in shaping more human-like policies across five distinct tasks. **They perform better**: in two image reconstruction tasks, PLLB agents generate instructions increasing the listeners' performance compared to non-linguistic baselines (Section 7), and PLLB agents also help more efficiently learn robot grasping policies (Section 8). **They generalize better**: in a maze task, PLLB helps uncover the abstract problem structure and generalize better to similar mazes (Section 6), while also reducing overreliance on non-generalizable visual features in robotic manipulation (Section 8). **They are more interpretable**: in a coordination task, agents converge on humans' preferred policy when multiple optimal policies exist (Section 5). **They are more inter-operable**: in maze and image reconstruction tasks, humans achieve better rewards when interacting with PLLB agents compared to agents trained without a bottleneck (Sections 6 and 7).

2 Background & Related Work

PLLB is inspired by the dual use of language as both a communicative and cognitive tool for decision-making in both humans (Carruthers & Boucher, 1998) and machines (Colas et al., 2022).

Language for communication. Language facilitates cooperation and coordination between humans and machines via instructions (Hermann et al., 2017; Chevalier-Boisvert et al., 2019), advice (Watkins et al., 2021), explanations (Zhong et al., 2020; Lampinen et al., 2022), or the formation of conventions (Hawkins et al., 2020; Hu & Sadigh, 2023). Such communicative functions increase the fidelity and breadth of cultural transmission — a process of social learning that underlies human ecological success (Mesoudi & Thornton, 2018). PLLB agents not only learn from language, but also generate their own to be shared with others.

Language for cognition. Language also augments a learner's cognitive abilities. Language-augmented RL agents represent more abstract goals (Jiang et al., 2019), generalize better (Hill et al., 2020; Colas et al., 2020; Wong et al., 2021), explore more efficiently (Colas et al., 2020; Tam et al., 2022; Klissarov et al.,

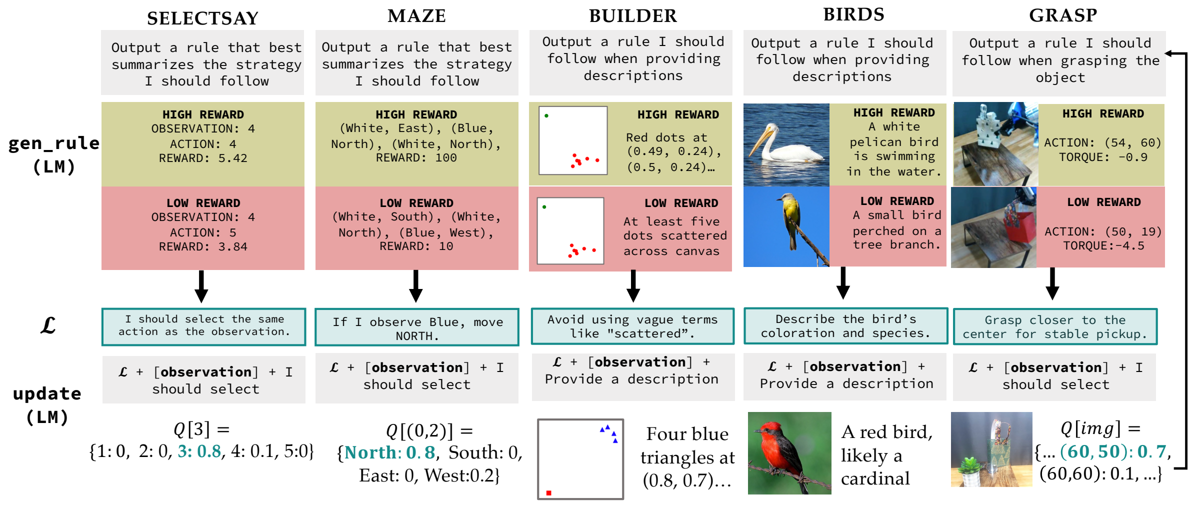

SELECTSAY
MAZE
BUILDER
BIRDS
GRASP
Output a rule that best summarizes the strategy I should follow
Output a rule that best summarizes the strategy I should follow
Output a rule I should follow when providing descriptions
Output a rule I should follow when providing descriptions
Output a rule I should follow when grasping the object
gen_rule (LM)
HIGH REWARD OBSERVATION: 4 ACTION: 4 REWARD: 5.42
HIGH REWARD (White, East), (Blue, North), (White, North), REWARD: 100
HIGH REWARD Red dots at (0.49, 0.24), (0.5, 0.24)…
HIGH REWARD A white pelican bird is swimming in the water.
HIGH REWARD ACTION: (54, 60) TORQUE: -0.9
LOW REWARD OBSERVATION: 4 ACTION: 5 REWARD: 3.84
LOW REWARD (White, South), (White, North), (Blue, West), REWARD: 10
LOW REWARD At least five dots scattered across canvas
LOW REWARD A small bird perched on a tree branch.
LOW REWARD ACTION: (50, 19) TORQUE:-4.5
$\mathcal{L}$
I should select the same action as the observation.
If I observe Blue, move NORTH.
Avoid using vague terms like "scattered".
Describe the bird's coloration and species.
Grasp closer to the center for stable pickup.
update (LM)
$\mathcal{L}$ + [observation] + I should select
$\mathcal{L}$ + [observation] + I should select
$\mathcal{L}$ + [observation] + Provide a description
$\mathcal{L}$ + [observation] + Provide a description
$\mathcal{L}$ + [observation] + I should select
$Q[3]$ = {1: 0, 2: 0, 3: 0.8, 4: 0.1, 5:0}
$Q[(0,2)]$ = {North: 0.8, South: 0, East: 0, West:0.2}
Four blue triangles at (0.8, 0.7)…
A red bird, likely a cardinal
$Q[img]$ = {… (60, 50): 0.7, (60,60): 0.1, …}

Figure 2: PLLB can be applied to a diversity of domains. In all domains, it iterates between `gen_rule`, a function that prompts an LM to extract a linguistic rule ($\mathcal{L}$, blue) by contrasting high and low reward episodes from past experience (green and red boxes), and `update`, a function for updating an agent's policy through interaction with the environment conditioned on $\mathcal{L}$. PLLB can be applied to multi-step decision making tasks and visual or robotics tasks alike with minimal implementation variations (replacing the LM by a VLM in `gen_rule`, or replacing policy regularization by simple instruction conditioning in `update`. Exact prompts are in Appendix Section B.7.

2023) and can decompose complex goals into simpler ones (Chen et al., 2021; Ahn et al., 2022; Hu et al., 2022; Sharma et al., 2021; Hu & Clune, 2023). Our work extends these benefits to agents that learn from *self-generated* linguistic feedback.

Inner speech. Generating linguistic rules for oneself is a form of *inner speech*. In the Vygotskian tradition, inner speech is seen as the internalization of the social speech generated by caretakers to help children solve problems (Vygotsky, 1965; Luria, 1959). As a result, it is thought to support our capacities for complex, long-term behaviors (Vygotsky, 1965; Luria, 1959; Hermer-Vazquez et al., 2001; Spelke, 2003). AI agents endowed with forms of inner speech (explanations, descriptions or subgoals) have been found to perform and generalize better than agents trained with purely neural representations (Wong et al., 2021; Lampinen et al., 2022; Roy et al., 2022; Hu & Clune, 2023; Kim et al., 2020). Unlike these approaches, PLLB approach generates language in an unsupervised way and maximizes downstream performance as well as interpretability and inter-operability.

Multi-step reasoning and text agents. Recent works have proposed guiding LMs' reasoning by prompting them to step through sequences of "thoughts" (Wei et al., 2023; Yao et al., 2022; Li et al., 2023b; Shinn et al., 2024). Our approach similarly uses a language bottleneck to concisely express intermediate information useful for later behavior (i.e. policy learning). However, unlike text-based reasoning agents such as ReAct (Yao et al., 2022), Rememberer (Zhang et al., 2023), and Reflexion (Shinn et al., 2024), **PLLB does not require the underlying policy to be fully expressible in text**. For example, we show that a text-only maze solving agent completely fails, whereas PLLB can improve classical Q-learning style agents (Section 6). Moreover, instead of describing actions for a particular state, PLLB rules capture high-level strategies over a sequence of states and actions, enabling generalization. We also show the learned rules can be shared to improve human task performance across multiple user studies (Sections 6 and 7).

3 The Language Bottleneck

PLLB builds on the standard RL framework to train agents to solve decision-making tasks in human-like ways. We formalize decision-making tasks (e.g., solving a maze) as Markov decision processes (S, A, f, R, T)

with reward function $R : S \times A \to \mathbb{R}$ (e.g., solving speed) over states S (e.g., cell coordinates) and actions A (e.g., directions), finite time horizon T, and a deterministic transition function $f : S \times A \to S$ that maps state–action pairs (s, a) to next states s (e.g., moving between cells). Standard RL then involves training a policy π to maximize expected reward (e.g., learning to solve the maze faster). This is usually done by alternating between two steps: (1) collecting data with the current policy: $D \leftarrow \pi_i$ and (2) updating the policy using the data: $\pi_{i+1} \leftarrow \texttt{update}(\pi_i, D)$, where `update` implements some RL algorithm (Sutton & Barto, 2018). This procedure often fails to yield interpretable, human inter-operable, or generalizable behaviors.

Our key idea is to introduce a *language bottleneck* between data collection and policy update. We extract linguistic rules explaining past rewarding behaviors and then use them to regularize the policy's behavior in the next learning iteration (e.g., *I should go right in blue cells*, see Figure 1). Our experiments will show that this improves the interpretability and inter-operability (communicative use), and performance and generalization (cognitive use) of our agents over standard RL baselines in a variety of tasks (see Figure 2 and Sections 5 to 7). The resulting algorithm thus alternates between three steps: (1) data collection, $D_i \leftarrow \pi_i$; (2) language bottleneck generation, $\mathcal{L}_i \leftarrow \texttt{gen\_rule}(D_i)$ and (3) policy updating, $\pi_{i+1} \leftarrow \texttt{update}(\pi_i, D_i, \mathcal{L}_i)$, see Figure 1. In PLLB, the first step is implemented just as in ordinary policy learning approaches; we next describe the last two steps in more detail.

3.1 Rule Generation (`gen_rule`)

Using all the experience D_i collected by the policy π_i in the current iteration, `gen_rule` aims to infer an abstract rule $\mathcal{L}_i$ that best explains the agents' successful behaviors: $\mathcal{L}_i \leftarrow \texttt{gen\_rule}(D)$. This is done by prompting an LM with contrastive episodes from D_i (top-N highest vs. top-N lowest total rewards) and asking it to *provide the rule that should be followed to obtain high rewards* (see first row of Figure 2 and Appendix Section B.7 for full prompts). Importantly, this requires the first iteration of `gen_rule` to start only once we observe a pair episodes with sufficiently different rewards. We found this contrastive approach, inspired by Zhong et al. (2023) and Dunlap et al. (2023), to provide more precise rules than simply summarizing high-reward strategies.

3.2 Rule-Guided Policy Update (`update`)

Given a rule $\mathcal{L}_i$, the `update` step produces a new policy $\pi_{i+1} \leftarrow \texttt{update}(\pi_i, D_i, \mathcal{L}_i)$ that is better aligned with $\mathcal{L}_i$. There exist many methods for leveraging language instructions, typically provided by experts, to modify agents' policies. One approach we leverage is InstructRL (Hu & Sadigh, 2023), which regularizes the learned policy with another policy induced by the linguistic rule $\pi_{\mathcal{L}}$. For instance, in the maze example shown in Figure 1, if the rule is *I should go right on every blue cell*, the induced policy $\pi_{\mathcal{L}}$ should assign probability 1 to the *right* action in blue cells, but equal probabilities to all actions in every other situations. In the Q-learning algorithm, this approach simply adds a *regularizing term* (orange) to the standard Q-learning update rule [2]:

$$Q^\theta(s_t, a_t) \leftarrow r_{t+1} + \gamma Q^\theta(s_{t+1}, a_{t+1}) \quad \text{where} \quad a_{t+1} = \arg\max_a \left[Q(s_{t+1}, a) + \lambda \log \pi_{\mathcal{L}}(a \mid s_t)\right].$$

Here γ is a discount factor and λ controls the strength of the rule-induced regularization and could be made time-dependent with a pre-defined or learned schedule (e.g., stronger regularization early).

But how do we induce $\pi_{\mathcal{L}}$ from $\mathcal{L}$? Since $\mathcal{L}$ is expressed in natural language, we may obtain a rule-conditional policy by prompting an LM. In particular, our experiments condition the LM on both the current rule and the current state of the agent s_t, and instruct the LM to generate the next action to obtain a probability distribution over admissible actions (e.g directions), as in Hu & Sadigh (2023). While these prompted LMs may perform tasks poorly on their own (e.g. because of their inability to perform long-range planning or process complex visual input), the regularizer may nonetheless guide Q-learning in the right direction. Running Q-learning updates with the regularization term from experience data gives us a new Q-table Q^θ_{i+1}

[2]This regularizing term can also be applied in the function approximation setting (Mnih et al., 2013).

from which we can derive the new policy π_{i+1} by taking actions with maximum expected value in every state $\pi_{i+1}(s_t) = \max_a Q^{\theta}_{t+1}(s_t, a)$.

Finally, there exist some domains where the policy can be directly implemented by an LM or VLM, which is leveraged by existing methods such as ReAct (Yao et al., 2022) or Rememberer (Zhang et al., 2023). In these cases, simply conditioning the policy on the rule $\mathcal{L}$ by adding it to the prompt $\pi_{\mathcal{L}}$ might be enough to steer the agent's behavior to become more human-like: $\pi_{i+1} \leftarrow \pi_{\mathcal{L}_i}$. Given a new policy π_{i+1}, we can now close the loop and generate a new rule $\mathcal{L}_{i+1}$.

This training paradigm is thus compatible with a variety of architectures for the target policy π. We implement PLLB with tabular RL agents in Sections 5 and 6, with LM/VLM policies in Section 7, and visuomotor robot controllers in Section 8.

4 Experiment Set-Up

We run experiments on a diverse set of five tasks, with different agent architectures and action spaces, to showcase how PLLB can train more human-like agents. These include a simple two-player communication game called SaySelect (Section 5), a Maze solving task (Section 6), two collaborative image reconstruction tasks with synthetic (Builder) and natural (Birds) images (Section 7), and grasp planning for a 7-DoF robot (Section 8). For all tasks, we include hyperparameter details (Appendix B.1), the exact prompts used in `gen_rule` and `update` (Appendix B.7), and examples of generated rules. For our main experiments, we prioritized using accessible and low-cost open weight language models such as llama-2-70b-chat. However, we analyze the effect of changing the underlying language model on `gen_rule` in Appendix B.8 and B.9.

To evaluate the *cognitive* function of PLLB, we first evaluate the policies learned by artificial agents in each environment based on performance metrics such as task reward, similarity to known human-like policies, or generalization. For all tasks, we compare our **Bottleneck** approach with baseline methods (e.g., standard RL, base LM) and an **Adversarial** ablation of PLLB that leverages invalid linguistic rules (e.g., obtained by labeling high reward examples as low reward) in `gen_rule`.

To evaluate the *communicative* function of PLLB, we conduct several human subject studies showing that generated rules $\mathcal{L}$ can help improve human task performance on a variety of metrics (e.g., reward, solving time). Participants were either recruited in-person (for Maze) or on the Prolific crowdsourcing platform[3] (Builder, Birds), compensated at a rate of US$12/hour. All studies were approved by our institution's IRB. We believe our work is the first to validate self-generated agent rules from LMs for training human participants.

5 SaySelect

As a proof-of-concept, we first consider SaySelect, a simple collaborative game introduced in Hu & Sadigh (2023). Here, a *speaker* can see a hidden set of five balls including two blue ones and must help a *listener* find the two blue balls by communicating with them only via numbers, see Appendix Figure 8. In principle, the speaker and listener could converge on any bijective convention associating messages to balls, and this is what we find with RL agents (multiple optimal policies). Humans, on the other hand, empirically prefer the mapping $1 \rightarrow 1$, $2 \rightarrow 2$, etc. which we call the *human-interpretable policy.* Whereas Hu & Sadigh (2023) showed that regularizing the listener's policy with the instruction *I should select the same number as my partner* successfully guided the two RL agents to the human-interpretable policy, we ask whether PLLB could learn a similar rule and achieve a similar interpretability *on its own*?

Task Overview. The speaker and listener are RL agents trained with Q-Learning that receive positive rewards when the listener collects blue balls, negative rewards otherwise. After training both agents for 200 episodes, we prompt llama-2-70b-chat[4] with the task description and a contrastive set of high- and low-reward episodes to generate a rule $\mathcal{L}$, e.g., *I should choose action 4 whenever the observation is 2, 3, 4, or 5.* (`gen_rule`, see Figure 2).

[3]Prolific, `http://prolific.com`.
[4]https://huggingface.co/meta-llama/Llama-2-70b-chat-hf

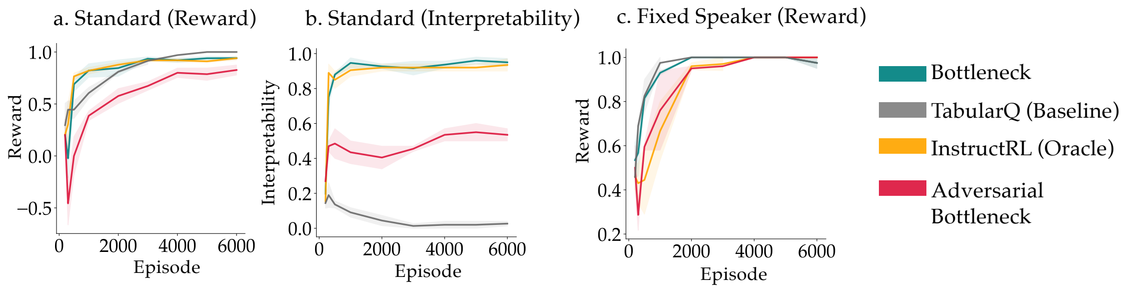


Figure 3: SaySelect. a) **Bottleneck** agents learn as fast as **TabularQ** and **InstructRL** and faster than agents using adversarial rules **Adversarial**. b) Unlike **TabularQ**, they learn human-interpretable policies; this without relying on an external instruction like **InstructRL**. c) When faced with speakers enforcing a non-human-interpretable policy, **Bottleneck** converges faster than **InstructRL**.

We generate new rules every 500 training episodes. In between, we regularize the listener's policy with a another policy $\pi_{\mathcal{L}}$ induced from the current rule $\mathcal{L}$. This is done by applying a *softmax* on the LM's logits (see method in Section 3) across all possible actions and obtaining a distribution. In the following experiments, we compare our **Bottleneck** method, with the **Adversarial** version (corrupted rule), a **TabularQ** baseline, as well as an **InstructRL** upper bound using the same predefined instruction as (Hu & Sadigh, 2023).

5.1 PLLB helps learn human-interpretable policies

We first compare the four methods along two dimensions: their reward and their *interpretability* measured as the proportional (0 to 1) similarity between the listener's policy and the optimal human-interpretable policy (the intuitive message-ball described above). All methods but **Adversarial** quickly solve the task (Figure 3a), showing that corrupted rules hinder learning. **Bottleneck** and **InstructRL** both converge on the human-interpretable policy thanks to their linguistic rules while **TabularQ** — deprived of such rule — does not (Figure 3b). Interestingly, **Adversarial** policies are more interpretable than **TabularQ** ones, suggesting that even corrupted language may provide inductive biases towards more human-like behavior. Qualitatively, we observe that generated rules converge towards describing the human-interpretable policy (e.g., *I should follow the strategy of choosing the same action as Agent 1*, see examples in Section B.8).

5.2 PLLB can learn counter-intuitive policies

A potential confound in the preceding experiment is the possibility that PLLB converges to a human-interpretable policy because this is the *only* behavior an LM can describe. To test this hypothesis, we fixed the speaker to use a counter-intuitive policy using a random mapping between balls and messages (e.g., $1 \rightarrow 3$, $5 \rightarrow 2$, etc.). Here, the human-interpretable policy is mis-aligned, and would lead the listener to fail or learn slower. In Figure 3, we observe that while **InstructRL** and **Adversarial**, both regularized by misaligned rules (the human-interpretable one for **InstructRL**, another random permutation for **Adversarial**), eventually adapt to the fixed speaker, they converge at a slower rate than both **TabularQ** (no rule) and **Bottleneck**. This effect was more pronounced at higher values of λ. Overall, **Bottleneck** is the only method that adapts to a fixed speaker policy while also converging on the human-like policy when the speaker is learning, without human supervision.

6 Maze

In the Maze domain, agents must navigate a maze to find the goal using four directional actions (N/S/E/W). We study mazes in which environment cues provide hints about the optimal path. This setup helps us study the impact of the language bottleneck: can PLLB agents infer the underlying structure and use it to generalize across mazes (cognitive use)? Can they transmit this knowledge to humans and help them perform better in novel settings (communicative use)?

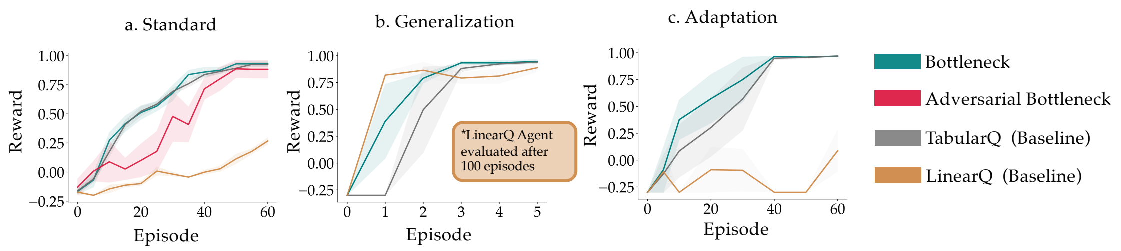


Figure 4: Results in Maze. a) **Bottleneck** agents learn as fast as the non-linguistic **Baseline agents**, but faster than **Adversarial** and **LinearQ** agents. b) When faced with a new maze with similar structure, **Bottleneck** agents learn faster than **TabularQ** (which does not perceive color) and **LinearQ**(which does, but learns slower). c) When faced with a maze of a different structure, **Bottleneck** agents adapt swiftly while **LinearQ** cannot recover. We do not evaluate **Adversarial** for the Generalization or Adaptation experiments due to its poor performance in the standard setting.

Task Overview. We randomly generate different 7x7 mazes with the gym-maze code base.[5] Agents receive as reward the inverse of the number of steps they needed to reach the goal as an incentive to learn the optimal solution. Although all mazes are random, we introduction additional structures by coloring cells for which the optimal action is *south* (red) or *north then east* (blue) with probability 50% (or they are left blank). Therefore, agents that leverage color information can better generalize to new mazes where the coloring is preserved, even if the optimal action sequence differs.

The first iteration of PLLB corresponds to training a standard RL algorithm to obtain π_1, in our case a tabular Q-learning agent (**TabularQ**). After the agent observes at two solved mazes, we run `gen_rule` by prompting a llama-2-70b-chat LM with contrastive episodes and a task description. This gives us a linguistic rule $\mathcal{L}_1$ (e.g., *Upon observing RED, take SOUTH*) that we can use to `update` the policy. We first induce the regularizing policy $\pi_{\mathcal{L}_1}$ with rule $\mathcal{L}_1$ by obtaining a probability distribution over the 4 actions from the LM, and then run the RL algorithm for 5 episodes to obtain the new policy π_2 (procedure described in Section 3). We repeat these steps every 5 episodes of interactions with the environment.

Baselines include the tabular Q-learning algorithm without language bottleneck (**TabularQ**) and a variant of **Bottleneck** generating rules from reward-randomized episode samples (**Adversarial**). We also evaluate **LinearQ**, an agent learning a linear model Q-function with an additional feature for cell color. We train **LinearQ** with a batch size of 10 and learning rate 0.001, after performing a hyperparameter sweep. We found **LinearQ** took significantly longer to converge than the other methods, likely due to the increased state representational complexity.

6.1 PLLB learns more generalizable policies

Figure 4a shows that, in the Standard setting, learning a valid rule (**Bottleneck**) does not increase learning speed of a single maze over **TabularQ**. However, using a corrupted rule (**Adversarial**) or learning from linear features (**LinearQ**) slows down learning. Furthermore, **Bottleneck** outperforms **TabularQ** with respect to generalization: when we switch to a new 7x7 maze sharing the same underlying color semantics, but with a different optimal action sequences. **Bottleneck** leverages the learned rule and generalized to the new maze more effectively than all other agents (Figure 4b). For a a fair comparison, we start with the fully converged policy for each method (requiring 100 episodes for **LinearQ**). While **TabularQ** cannot generalize because it does not perceive colors, **LinearQ** does generalize faster at first, but converges slower.

The generated rules improve over time to better capture the underlying structure of the maze (e.g., *if I observe BLUE, then take the NORTH action*; see other examples in Appendix B.8). Across all trials, 100% of the final rules mention the red $\rightarrow$ *south* rule and 60% uncovered the more complex blue $\rightarrow$ *north-then-east* rule. Finally, we find that a policy that does not update Q-values (that is, purely implemented via the LM) achieves a success rate 0% within the same time limit, even when using the final rules generated from

[5]https://github.com/MattChanTK/gym-maze

PLLB. This highlights the importance of leveraging experience that cannot be expressed with language, which text-based reasoning agents cannot do.

6.2 PLLB learns adaptable policies

In Figure 4c, we reproduce this same experiment on a maze with a different underlying structure (red now indicates *west* while blue indicates *east then south*). Although the rule **Bottleneck** learned does not apply anymore, it can still adapt faster than the baselines. We find that all trials end with rules capturing the new mapping (100% red → *west*, 50% blue → *east-then-south*). Meanwhile, **LinearQ** overfits to the first maze and struggle to adapt. Overall, our results show that PLLB supports human-like cognitive functions by finding a tradeoff between the efficiency of TabularQ and the generalization capabilities LinearQ, while remaining more adaptable.

6.3 PLLB is more interpretable and inter-operable

Can generated rules be useful to humans as well? We asked 50 university students to solve a 7x7 maze in the fewest steps possible using arrow keys. They could only observe the cells they had already visited and the walls they had already bumped into (see Figure 10 in Appendix). We split them into three groups: a **Control** group receiving no assistance, and two others receiving information about a *similar* but different maze sharing the same underlying color semantics. The **Visual** group is shown a visual representation of the optimal policy (arrows in each cell, see Figure 9 in Appendix), while the **Bottleneck** group is provided a randomly sampled language rule $\mathcal{L}$ learned by PLLB. This set-up lets us evaluate how generalizable the two different aids are to a new maze, as well as how quickly participants can account for incorrect information depending on modality.

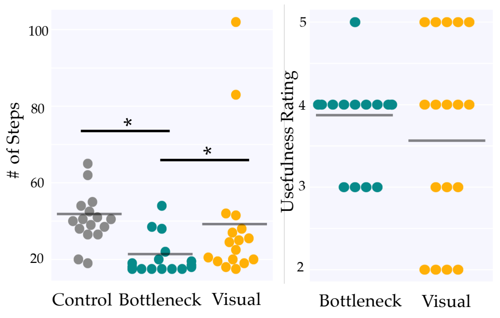


Figure 5: Participants given PLLB rules solve new mazes faster than those given either visual or no aid (control), left. They also self-reported linguistic feedback as more useful than visual (right). * marks statistical significance at level 0.05 with two-sided Mann-Whitney test after Bonferroni correction.

Participants using PLLB rules solve the new maze with fewer steps than others and find this aid significantly more useful than the visual one in average (see Figure 5). Participants indeed found it harder to extract the relevant information from the visual aid, which contained a great deal of non-transferable information (optimal actions in all non-colored cells). PLLB rules focus on the useful and transferable information learned by the previous agent, which is much more readily usable by humans. Overall, the results in Maze demonstrate the ability of PLLB to train agents that are more generalizable and adaptable (cognitive use) as well as more interpretable by, and inter-operable with humans (communicative use).

7 Collaborative Image Reconstruction

PLLB is not restricted to the training of RL agents. This section introduces two collaborative image reconstruction tasks inspired by the collaborative assembly task of McCarthy (2020). Builder and Birds both consider two agents: a *speaker*, who can see a hidden target image, and a *listener*, who must accurately reconstruct the target image based on a description from the speaker. Both agents aim to converge to a description style leading to high reconstruction accuracy. We consider both synthetic images built from sequences of discrete actions by the listener (Builder), and natural images of birds that the listener reconstructs using a text-to-image generation model (Birds). Here, PLLB combines with state-of-the-art multi-modal models to let agents perceive semantic visual features.

7.1 Task Overview

Dataset. In Builder, we construct a dataset of synthetic images with a variable number of shapes (triangle, square, circle) in different colors (magenta, blue, red, green) and locations on a 2D grid. Each image

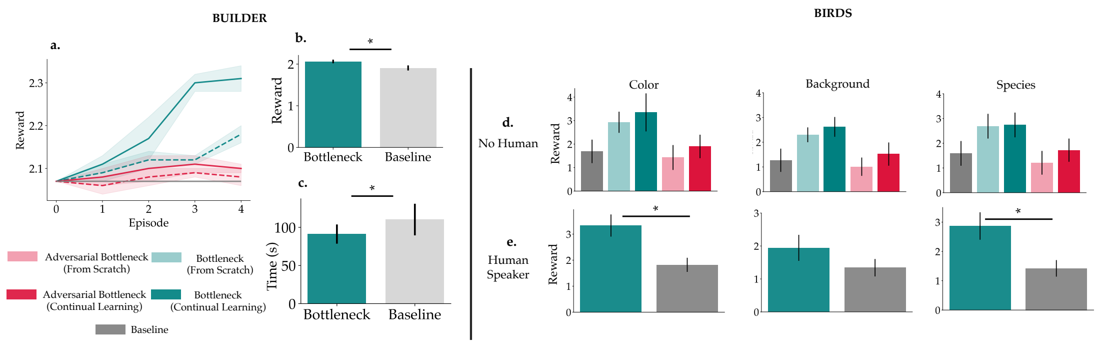


Figure 6: BUILDER: **Bottleneck** listeners improve upon **Baseline** descriptions over time, unlike an (**Adversarial**) listener that inaccurately represents rewards. Human users reconstructed target images with b. higher accuracy, and c. less time, when provided **Bottleneck** instruction vs. original **Baseline** descriptions. BIRDS: PLLB rules (3 iterations) help d. automated and e. human speakers generate better descriptions for naturalistic image reconstruction for three distinct reward functions.

is created by a sequence of discrete actions, each representing a particular combination of shape, color, and location (e.g., `[ACT12 ACT2 ACT4]`, where ACT2 adds a magenta circle in the lower right area of the grid). In BIRDS, we construct a dataset of natural bird images by selecting 5 images for each of 10 bird species from the CUB-200-2011 dataset provided by Wah et al. (2011) (see Appendix Section B.4 for species list).

Reward. In BUILDER, since target and reconstructed images are fully determined by their action sequences, we measure task success via the Levenshtein similarity between the (sorted) corresponding sequences. In contrast, the BIRDS listener outputs text-to-image generations with varied properties, so we define three reward functions: **Color**, **Background**, and **Species**. To evaluate a reconstructed image x_r against a target x_t, we form a *contrast* set C from three CUB-200-2011 images differing in the target property (e.g., color) but most similar in others. Reward is then $r = \sum_{c \in C} d(\mathsf{CLIP}(x_r), \mathsf{CLIP}(x_c)) - d(\mathsf{CLIP}(x_r), \mathsf{CLIP}(x_t))$, where CLIP denotes the CLIP image embedding (Radford et al., 2021). For both BUILDER and BIRDS, rewards are reported on a held-out validation set.

Models. As open-source VLMs struggle with describing synthetic images, we use the llama-2-70b-chat LM as our speaker for BUILDER, representing images in raw text that list all shapes' type, color, and exact coordinates individually in sequence. On the other hand, we use the open source Llava VLM as our speaker for BIRDS (Liu et al., 2023). We prompt both speaker models using the target image and a general task description, as well generated rules after the first iteration.

We implement the listener for BUILDER with a neural sequence-to-sequence model pre-trained on English text (Lewis et al., 2019), which we fine-tune at each episode on a training set of (description, action sequence) pairs using descriptions provided by the speaker. Meanwhile, we use the Stable Diffusion text-to-image diffusion model as the listener for BIRDS.

For both tasks, we consider two settings: **From Scratch**, where the listener is initialized using the original pre-trained model weights at every episode, and **Continual Training**, where the listener is continuously updated over time using descriptions by the speaker. We create separate held-out splits for training, early stopping, selecting samples for `gen_rule`, and evaluation.

Implementation. We first generate base image descriptions for both tasks by providing the speaker a prompt containing a general task description and target image, but without any rule $\mathcal{L}$. For the BUILDER task, an original image description is: *At least five dots in total - four red and one green - scattered across the canvas, with two sets of matching locations (x = 0.49 / y = 0.24 and x = 0.5 / y = 0.24), another set at x = 0.52 / y = 0.3, and finally, one dot located near x = 0.76 / y = 0.71.* This description is overly complex, mixes different coordinate formats, and difficult for the listener to use to reconstruct the target image, leading to low reward. Meanwhile, an example original image description for the BIRD task is: *"a* *bird on a branch"*. While simple, this description does not contain sufficient information about the **Color**, **Background**, and **Species**, causing the listener to generate an image reconstruction with low reward.

We next evaluate these base descriptions using each task's respective reward functions, and select the (image, description) examples with the 5 highest and lowest rewards. Using the prompts shown in Section B.7, we implement `gen_rule` using the `llama-2-70b-chat` and `llava` models for Builder and Birds, respectively. For Birds, we observe that output rules $\mathcal{L}$ are specific to the reward function: an example rule we get for the **Species** reward is: *"Identify the bird's species if possible, and include any distinctive characteristics that set it apart from other birds."*, while an example rule we get for the **Background** reward is: *"Describe the setting or background of the image, such as the presence of snow, water, or other elements"*.

We implement `update` for both tasks by simply appending the output rule $\mathcal{L}$ to the original prompt to the speaker, as shown in Figure 2, using the speaker's output as the new description for a given target image. We repeat this cycle of eliciting rule $\mathcal{L}$, appending $\mathcal{L}$ to the same prompt used to generate the base instructions in order to re-label our data with new instructions, and training and evaluating the listener model for a total of 5 iterations for Builder and 3 iterations for Birds. We qualitatively show this leads to improved image reconstruction from the listener in Appendix Figure 13, and next discuss our experimental results.

7.2 PLLB helps speakers provide more usable instructions

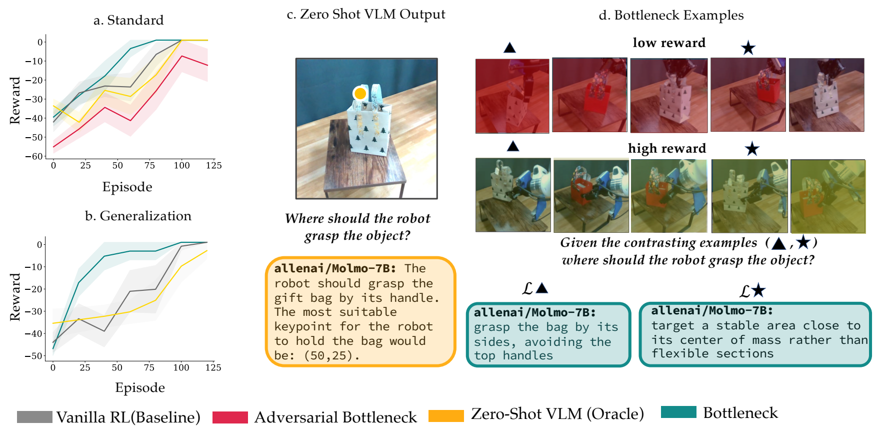
For both tasks, listeners following rules generated by PLLB (**Bottleneck**) outperform listeners trained from uninformed (**Baseline**) or misinformed (**Adversarial**) speakers (Figure 6). This holds true for all three different reward functions in Birds, as well as both training settings, with **Continual Learning** learning enabling **Bottleneck** to have an even stronger improvement over **Baseline** image descriptions by leveraging learned task experience. **Continual Learning** does not significantly help the **Adversarial** method, emphasizing that the linguistic rules $\mathcal{L}$ in **Bottleneck** do capture succesfull task strategies. However, **Continual Learning** is not required for **Bottleneck** to outperform **Baseline** image descriptions. The success of the **From Scratch** training setting can be interpreted as the speaker generating abstract rules to guide its own learning, leading to improved performance and showing evidence for a cognitive use of language.

Finally, we observe that the generated rules $\mathcal{L}$ encourage the speaker to reduce vague and general language, and describe properties relevant to the corresponding underlying reward function in Birds, although this might take several cycles to fully converge (see example rules in Appendix Section B.8). Furthermore, rules $\mathcal{L}$ become more complex over time, and across different trials we observe path dependency where different trials converge to different rules (e.g., (x, y) vs. $x = 0.xx, y = 0.yy$ for coordinates in Builder). Overall, these rules help guide the speaker and listener agents in both tasks to iteratively improve image reconstruction over time, which can also be seen qualitatively in Appendix Figure 13.

7.3 PLLB can collaborate with human listeners

Do human listeners benefit from image descriptions provided by speakers following **Bottleneck** rules? We conduct a human subject study for the Builder task, where we replace the listener with study participants at the end of one iteration of PLLB, and evaluate how quickly they can reconstruct target images from a held-out test set. We recruit 20 crowdworkers on Prolific to reconstruct 5 images using descriptions from a speaker following a rule $\mathcal{L}$ generated with (**Bottleneck**), and 5 images using the original (**Baseline**) descriptions. We provided participants an interface that included a drawing canvas and buttons controlling shapes and colors (Section B.3), allowing us to use Levenshtein similarity between (sorted) user actions and the underlying action sequences for a target image as reward.

Participants using **Bottleneck** descriptions built target images faster and more accurately than participants using control descriptions (Wilcoxon signed-rank test, $p < 0.05$, Figure 6). 55% preferred **Bottleneck** descriptions when asked in a post-study survey, describing them as *"more direct and less ambiguous"*. 10% found both descriptions types equally easy to follow while remaining participants preferred **Baseline** descriptions because they *they gave more flexibility*, showing a preference not captured in our reward function.



```
allenai/Molmo-7B: The robot should grasp the gift bag by its handle. The most suitable keypoint for the robot to hold the bag would be: (50,25).
```

```
allenai/Molmo-7B: grasp the bag by its sides, avoiding the top handles
```

```
allenai/Molmo-7B: target a stable area close to its center of mass rather than flexible sections
```

Figure 7: Results in Grasp. a) **Bottleneck** agents learn faster than the non-linguistic RL **Baseline agents**, as well as **Adversarial** and an **Oracle** zero-shot prompt of a VLM without any internal policy update. b) **Bottleneck** agents generalize faster than both **Baseline** (which over-relies on image details) and **Oracle** to unseen bag designs and locations, (c) shows the fixed linguistic rule d) Example grasps, shaded by reward.

7.4 PLLB collaborates with human speakers

Can humans benefit from rules generated by PLLB? We asked 12 Prolific crowdworkers to act as speakers and provide descriptions for our dataset of Birds images. Half of the participants were provided rules corresponding to one of the three reward functions ($\mathcal{L}_3$) while the other half were not. Figure 6 (bottom) shows that listeners instructed by human speakers provided with PLLB rules outperform those instructed by uninformed human speakers (**Baseline**) (Bonferroni-corrected t-tests $p < 0.05$ for **species** and **colors**).

The descriptions from users not provided a rule are more diverse and less focused e.g., *A barbed wire is an uncomfortable stop for a bird.* With PLLB rules on the other hand (e.g., *Describe the bird's coloration accurately*), they generate more specific descriptions: *Red and black bird on barbed wire. Bright red chest, red at top of head, black wings and beak* for the same target image (see other example rules in Appendix Table 4). PLLB agents can easily transmit what they learned from experience (the rule) to humans.

8 Robot Grasp Planning

Can PLLB rules improve policy learning in embodied tasks with complex action spaces such as robot planning? Existing works on LM-based agents (e.g., ReAct (Yao et al., 2022) or Reflexion (Shinn et al., 2024)) and robot policies (e.g., Code as Policies) leverage pre-trained models for action generation, but cannot be applied in domains where a (non-LM-based) policy must be updated using real-world RL signals.

Consider training a robot to lift a paper bag. The most intuitive grasp would be at the bag's handles; indeed, a zero-shot **Oracle** prompt of the molmo-7b multi-modal language model outputs grasp keypoints on bag handles (Figure 7). However, if the bag is too heavy, a robot should adapt to less common grasps (e.g. by the side) to avoid tearing. Because VLMs are not grounded with actual physical quantities, such adaptation is impossible without leveraging an external reward (Gao et al., 2024). On the other hand, existing RL-based approaches for robotic manipulation are often inefficient (Kalashnikov et al., 2018). We show that PLLB rules can be used to efficiently learn non-intuitive robot grasping policies.

8.1 Task Overview

In the Grasp task, our agent is a Franka Emika Panda Arm robot that needs to lift and move a paper bag filled with unknown objects off of a shelf, as shown in Appendix Figure 12. Due to the complex action space, we follow Kalashnikov et al. (2018) and decompose the robot's policy into (i) perceiving the scene and identifying an appropriate grasp location, and (ii) planning a path to this location.

We begin with a *fixed* keypoint-conditioned path-planning policy which is pre-trained on a dataset of 50 trajectories with 3D-point cloud observations using Sphinx (Sundaresan et al., 2025). This model outputs a trajectory of 7-DoF end-effector poses (x/y/z/yaw/pitch/row/gripper) for the robot to execute given the keypoint. We then use PLLB to train (via RL) a grasp prediction policy that selects a grasp keypoint (a 2-D position) given an image of the environment. We define a successful grasp as one where the robot can lift the bag without it breaking, and so set our reward to be 0 for a successful grasp, -100 if the bag is not grasped at all, and use the normalized downward torque (e.g. -20Nm) applied to the robot's shoulder joint as a dense reward for unsuccessful grasps. Initially, the agent performs random grasps. After it observes one successful grasp, we run `gen_rule` with the open-source MOLMO-7B vision–language model with contrastive images of successful and unsuccessful grasp keypoints. This gives us a linguistic rule $\mathcal{L}$ (e.g. *grasp the left side of the bag*) that we can use to `update` the policy. We regularize the initial grasping policy toward the policy $\pi_{\mathcal{L}}$ induced from rule $\mathcal{L}$ by prompting molmo-7b with $\mathcal{L}$, then instructing it to generate random keypoints.

We compare our **Bottleneck** method with an **Adversarial** version (swapping high and low reward grasps), a **Vanilla RL** baseline that only uses image observation data, and an **Oracle** that always guides the policy with the rule generated from zero-shot prompting molmo-7b (no constrasting examples based on a reward).

8.2 PLLB can learn counter-intuitive policies

As shown in Figure 7, when averaged across 20 random seeds, our **Bottleneck** converges to higher reward compared to all other methods. Because the **Oracle** method suggests a rule encouraging the policy to predict key points near the bag's handles, it requires more episodes to learn that this behavior leads to lower reward. Qualitatively, we observe the generated rules converge towards guiding the robot to grasp at less common locations (e.g., *Grasp the bottom of the object*, see examples in Appendix Section B.8). This shows that PLLB can help improve over standard RL methods even for learning counter-intuitive policies.

8.3 PLLB learns more generalizable policies

To evaluate generalization, we create an environment with a new bag, location, and set of visual distractions (see Appendix 12). As shown in Figure 7, the rules learned by **Bottleneck** outperform alternative methods for generalizing to this new environment. Because the **Vanilla RL** policy can only rely on image features, it is not able to leverage the more generalizable strategy PLLB rules capture. Overall, the GRASP results show that PLLB can improve policy convergence even for the complex dynamics of embodied tasks.

9 Limitations & Discussion

PLLB helps agents becomes more interpretable by and inter-operable with humans. In our current implementation, `rule_gen` requires converting episodes into LM-compatible representations (text/images), which may limit use with long-horizon sensorimotor trajectories. Nevertheless, our Grasp task successfully uses pretrained robotic models, and advances in long-context modeling can address this limitation. Rule generation can also fail when the LM abstracts too much (e.g., in BIRDS we found the rule: *"Use descriptive language that conveys the mood, such as the serenity of a snowy day or the freedom of flight"*). Improvement in multi-modal models and contrastive example selection can help mitigate these issues (Appendix B.9).

One interesting direction for future work is applying PLLB to tasks with complex reward functions that reflect hard-to-articulate human preferences, such as image captioning for accessibility (Nie et al., 2020) and personalized language models (Li et al., 2023a). Linguistic rules can enhance transparency and predictability even when users struggle to express their goals. Additionally, PLLB could generate not just abstract rules but also explanations, goals, or learning strategies, enabling cultural transmission of novel uses in `update` functions, such as mapping language to state/action abstractions or reward functions. This paves the way for more human-like AI agents that creatively integrate language with decision-making (Colas et al., 2022).

References

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. 2022.

Allen Institute for AI. Molmo blog. `https://molmo.allenai.org/blog`. Accessed: 2025-02-25.

Bastien Boutonnet and Gary Lupyan. Words jump-start vision: A label advantage in object recognition. *Journal of Neuroscience*, 35(25):9329–9335, 2015. doi: 10.1523/JNEUROSCI.5111-14.2015. URL `https://www.jneurosci.org/content/35/25/9329`.

Peter Carruthers and Jill Boucher. *Language and Thought*. Cambridge University Press, 1998.

Valerie Chen, Abhinav Gupta, and Kenneth Marino. Ask your human: Using human instructions to improve generalization in reinforcement learning. *Proc. of ICLR*, 2021.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. Babyai: A platform to study the sample efficiency of grounded language learning. *Proc. of ICLR*, 2019.

Sahil Chopra, Michael Henry Tessler, and Noah D Goodman. The first crank of the cultural ratchet: Learning and transmitting concepts through language. *Proc. of CogSci*, 2019.

Andy Clark. Magic Words: How Language Augments Human Computation. In Peter Carruthers and Jill Boucher (eds.), *Language and Thought*. Cambridge University Press, 1 edition, 1998. ISBN 978-0-521-63108-2 978-0-521-63758-9 978-0-511-59790-9.

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter F. Dominey, and Pierre-Yves Oudeyer. Language as a cognitive tool to imagine goals in curiosity driven exploration. *Proc. of NeurIPS*, 2020.

Cédric Colas, Tristan Karch, Clément Moulin-Frier, and Pierre-Yves Oudeyer. Language and culture internalization for human-like autotelic ai. *Nature Machine Intelligence*, 4(12):1068–1076, 2022.

Lisa Dunlap, Yuhui Zhang, Xiaohan Wang, Ruiqi Zhong, Trevor Darrell, Jacob Steinhardt, Joseph E. Gonzalez, and Serena Yeung-Levy. Describing differences in image sets with natural language, 2023.

Jensen Gao, Bidipta Sarkar, Fei Xia, Ted Xiao, Jiajun Wu, Brian Ichter, Anirudha Majumdar, and Dorsa Sadigh. Physically grounded vision-language models for robotic manipulation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024. URL `https://arxiv.org/abs/2309.02561`.

Dedre Gentner and Jeffrey Loewenstein. *Relational Language and Relational Thought*. Erlbaum, 2002.

Robert D Hawkins, Minae Kwon, Dorsa Sadigh, and Noah D Goodman. Continual adaptation for efficient machine communication. *Proc. of ACL*, 2020.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. Grounded language learning in a simulated 3d world. 2017.

Linda Hermer-Vazquez, Anne Moffet, and Paul Munkholm. Language, space, and the development of cognitive flexibility in humans: The case of two spatial memory tasks. *Cognition*, 79(3):263–299, 2001.

Mary Hesse. The Cognitive Claims of Metaphor. *The Journal of Speculative Philosophy*, 1988.

Felix Hill, Andrew K. Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L. McClelland, and Adam Santoro. Emergent systematic generalization in a situated agent. *Proc. of ICLR*, 2020.

Hengyuan Hu and Dorsa Sadigh. Language instructed reinforcement learning for human-AI coordination, 2023.

Hengyuan Hu, David J Wu, Adam Lerer, Jakob Foerster, and Noam Brown. Human-AI coordination via human-regularized search and learning, 2022.

Shengran Hu and Jeff Clune. Thought cloning: Learning to think while acting by imitating human thinking. *Proc. of NeurIPS*, 36, 2023.

Yiding Jiang, Shixiang Gu, Kevin Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. *Proc. of NeurIPS*, 2019.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, and Sergey Levine. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018. URL `https://arxiv.org/abs/1806.10293`.

Jinkyu Kim, Suhong Moon, Anna Rohrbach, Trevor Darrell, and John Canny. Advisable learning for self-driving vehicles by internalizing observation-to-action rules. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9658–9667, 2020. doi: 10.1109/CVPR42600.2020.00968.

Martin Klissarov, Pierluca D'Oro, Shagun Sodhani, Roberta Raileanu, Pierre-Luc Bacon, Pascal Vincent, Amy Zhang, and Mikael Henaff. Motif: Intrinsic motivation from artificial intelligence feedback. 2023.

George Lakoff and Mark Johnson. *Metaphors We Live By.* University of Chicago press, 2008.

Andrew K. Lampinen, Nicholas A. Roy, Ishita Dasgupta, Stephanie C. Y. Chan, Allison C. Tam, James L. McClelland, Chen Yan, Adam Santoro, Neil C. Rabinowitz, Jane X. Wang, and Felix Hill. Tell me why! – explanations support learning of relational and causal structure. *Proc. of ICML*, 2022.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, 2019.

Belinda Z Li, Alex Tamkin, Noah Goodman, and Jacob Andreas. Eliciting human preferences with language models. 2023a.

Chengshu Li, Jacky Liang, Andy Zeng, Xinyun Chen, Karol Hausman, Dorsa Sadigh, Sergey Levine, Li Fei-Fei, Fei Xia, and Brian Ichter. Chain of code: Reasoning with a language model-augmented code emulator. 2023b.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2023.

Jelena Luketina, Nantas Nardelli, Gregory Farquhar, Jakob Foerster, Jacob Andreas, Edward Grefenstette, Shimon Whiteson, and Tim Rocktäschel. A survey of reinforcement learning informed by natural language. 2019.

Aleksander R Luria. The directive function of speech in development and dissolution. *Word*, 15(2), 1959.

William McCarthy. Emergence of compositional abstractions in human collaborative assembly. 2020. URL `https://api.semanticscholar.org/CorpusID:233179096`.

Reid McIlroy-Young, Siddhartha Sen, Jon Kleinberg, and Ashton Anderson. Aligning superhuman ai with human behavior: Chess as a model system. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*, KDD '20. ACM, August 2020. doi: 10.1145/3394486.3403219. URL `http://dx.doi.org/10.1145/3394486.3403219`.

Alex Mesoudi and Alex Thornton. What is cumulative cultural evolution? *Proceedings of the Royal Society B*, 285(1880):20180712, 2018.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

Allen Nie, Reuben Cohn-Gordon, and Christopher Potts. Pragmatic issue-sensitive image captioning, 2020.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. Generative agents: Interactive simulacra of human behavior, 2023.

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

Nicholas A Roy, Junkyung Kim, and Neil Rabinowitz. Explainability via causal self-talk. *Advances in Neural Information Processing Systems*, 35:7655–7670, 2022.

Pratyusha Sharma, Antonio Torralba, and Jacob Andreas. Skill induction and planning with latent language. *Proc. of ACL*, 2021.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Proc. of NeurIPS*, 2024.

Elizabeth S Spelke. What makes us smart? core knowledge and natural language. *Language in mind: Advances in the study of language and thought*, pp. 277–311, 2003.

Priya Sundaresan, Hengyuan Hu, Quan Vuong, Jeannette Bohg, and Dorsa Sadigh. What's the move? hybrid imitation learning via salient points. In *Proceedings of the International Conference on Learning Representations (ICLR)*, February 2025.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction.* MIT press, 2018.

Allison Tam, Neil Rabinowitz, Andrew Lampinen, Nicholas A Roy, Stephanie Chan, DJ Strouse, Jane Wang, Andrea Banino, and Felix Hill. Semantic exploration from language abstractions and pretrained representations. *Proc. of NeurIPS*, 2022.

Michael Henry Tessler, Jason Madeano, Pedro A. Tsividis, Brin Harper, Noah D. Goodman, and Joshua B. Tenenbaum. Learning to solve complex tasks by growing knowledge culturally across generations, 2021.

L.S Vygotsky. *Thought and Language.* MIT Press, 1965.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset. Technical report, 2011.

Tony T. Wang, Adam Gleave, Tom Tseng, Kellin Pelrine, Nora Belrose, Joseph Miller, Michael D. Dennis, Yawen Duan, Viktor Pogrebniak, Sergey Levine, and Stuart Russell. Adversarial policies beat superhuman go ais, 2023.

Olivia Watkins, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Jacob Andreas. Teachable reinforcement learning via advice distillation. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Proc. of NeurIPS*, 2021.

Sandra R Waxman. The development of an appreciation of specific linkages between linguistic and conceptual organization. *Lingua*, 92:229–257, 1994.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

Catherine Wong, Kevin Ellis, Joshua B. Tenenbaum, and Jacob Andreas. Leveraging language to learn program abstractions and search heuristics. *Proc. of ICML*, 2021.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. 2022.

Hanako Yoshida and Linda B Smith. Sound symbolism and early word learning in two languages. In *Proc. of CogSci*, 2003.

Danyang Zhang, Lu Chen, Situo Zhang, Hongshen Xu, Zihan Zhao, and Kai Yu. Large language models are semi-parametric reinforcement learning agents. *arXiv preprint arXiv:2306.07929*, 2023.

Ruiqi Zhong, Peter Zhang, Steve Li, Jinwoo Ahn, Dan Klein, and Jacob Steinhardt. Goal driven discovery of distributional differences via language descriptions, 2023.

Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. RTFM: Generalising to new environment dynamics via reading. In *Proc. of ICLR*, 2020.

A Appendix

B Appendix

B.1 Hyperparameters

1. SaySelect: We use the same default parameters used in InstructRL (Hu & Sadigh, 2023), including setting the regularization strength $\lambda = 0.25$. Additionally, each time we invoke `gen_rule`, we create an ensemble of 3 rules, which we aggregate over when construct the probability distribution over actions during `update`.

2. Maze: We use the InstructRL objective with a tabular Q-learning agent, but introduce a ϵ_{LM} parameter that controls whether the regularization strength λ is 0 or 1 at each time-step in an episode. Although we did not observe a strong effect on modifying ϵ_{LM}, we did not explore values larger than $\epsilon_{LM} = 0.4$ as that led to increased inference cost and experiment latency. Finally, each time we invoke `gen_rule` we create an ensemble of 4 rules, which we aggregate over when construct the probability distribution over actions during `update`.

3. Builder: For the listener agent we finetune BART for a maximum of 100 epochs at each iteration of PLLB, employing early stopping on a held-out validation set and using the default arguments provided by the HuggingFace Transformers library (Lewis et al., 2019). Each time we invoke `gen_rule`, we same 3 rules and select the rule with the highest aggregate likelihood across all tokens. Likewise, for each image description (the speaker's action), we sample 3 possible descriptions under the given rule and select the one with the highest probability.

4. Birds: For the fine-tuned version of the listener agent, we finetune StableDiffusion for 1000 steps on a separate finetuning dataset. When generating images, we simply sample one image per description, using 10 inference steps and a guidance scale of 7.5.

5. Grasp: For grasp keypoint selection,we train an upper confidence bound agent on a neural contextual bandit using image features as context, with learning rate 0.01, with an $(100, 100)$ action space containing integer coordinates of the environment image. During `update`, we sample 20 coordinates following a given rule from `gen_rule`, which we use as the probability distribution over actions, setting $\lambda = 0.2$ as the regularization strength. To execute the grasp, we use a grasp model built by training `Sphinx` (Sundaresan et al., 2025) over 50 waypoint trajectories with learning rate 0.0001 for 100 epochs.

B.2 SelectSay Overview

B.3 Maze

See Figures 9 and 10.

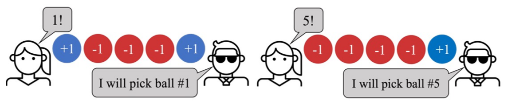


Figure 8: Overview of SaySelect game, reproduced from Hu & Sadigh (2023).

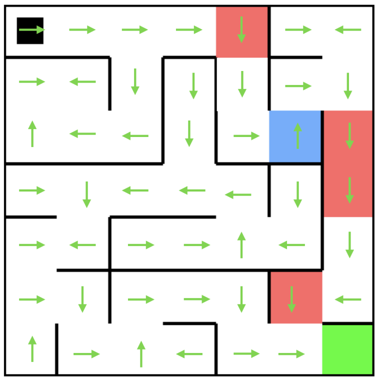


Figure 9: Visual aid provided to participants in the human subject study for the Maze task.

B.4 Dataset Details for Birds.

We consider images of the following bird species in the CUB-200-2011 dataset from Wah et al. (2011) for the Birds task: `[Indigo Bunting, Cardinal, Yellow Breasted Chat, American Crow, Vermillion Flycatcher, California Gull, Blue Jay, Tropical Kingbird, White Pelican, Horned Puffin]`.

B.5 Grasp

B.6 Builder and Birds Iterations

B.7 Full Prompts

We first provide the full prompts used for `gen_rule`. For space constraints, we do not include example `samples`, but note they follow the format shown in Figure 2.

1. SaySelect: You will be given a list of (OBSERVATION, ACTION, REWARD) examples collected from two agents learning to solve a task. Possible ACTIONS an agent can take are: 1, 2, 3, 4, 5, and quit. Each OBSERVATION describes the ordered sequence of actions that AGENT 1 picks, and each ACTION describes the ACTION that AGENT 2 picks based on the given OBSERVATION. The examples are separated into HIGH REWARD and LOW REWARD examples.+ `[samples]` +Output a language rule that best summarizes the strategy AGENT 2 should follow to receive HIGH REWARD, not LOW REWARD, based on the examples. Start the instruction with the prefix 'I should'.

2. Maze: You will be given a list of example (OBSERVATION, ACTION) trajectories collected from an AGENT learning to solve a maze. Each trajectory receives a REWARD. Possibles OBSERVATIONS an agent see are: WHITE, RED, BLUE Possible ACTIONS an agent can take are: NORTH, SOUTH, EAST, WEST. The examples are separated into HIGH REWARD and LOW REWARD examples + `[samples]` + Output a language rule that best summarizes the strategy the AGENT should follow when picking a sequence of ACTIONS to solve the maze and receive HIGH REWARD, not LOW REWARD, based on the examples. Start the instruction with the prefix 'I should'.

3. Builder: There are two agents. The goal of Agent 1 is to provide instructions to Agent 2 that helps Agent 2 to successfully recreate the image. You will be given a list of (ORIGINAL, AGENT 1 INSTRUCTION, REWARD) values where ORIGINAL is the original description of an image, INSTRUCTION is the instruction

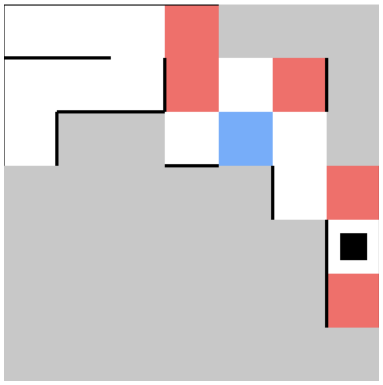
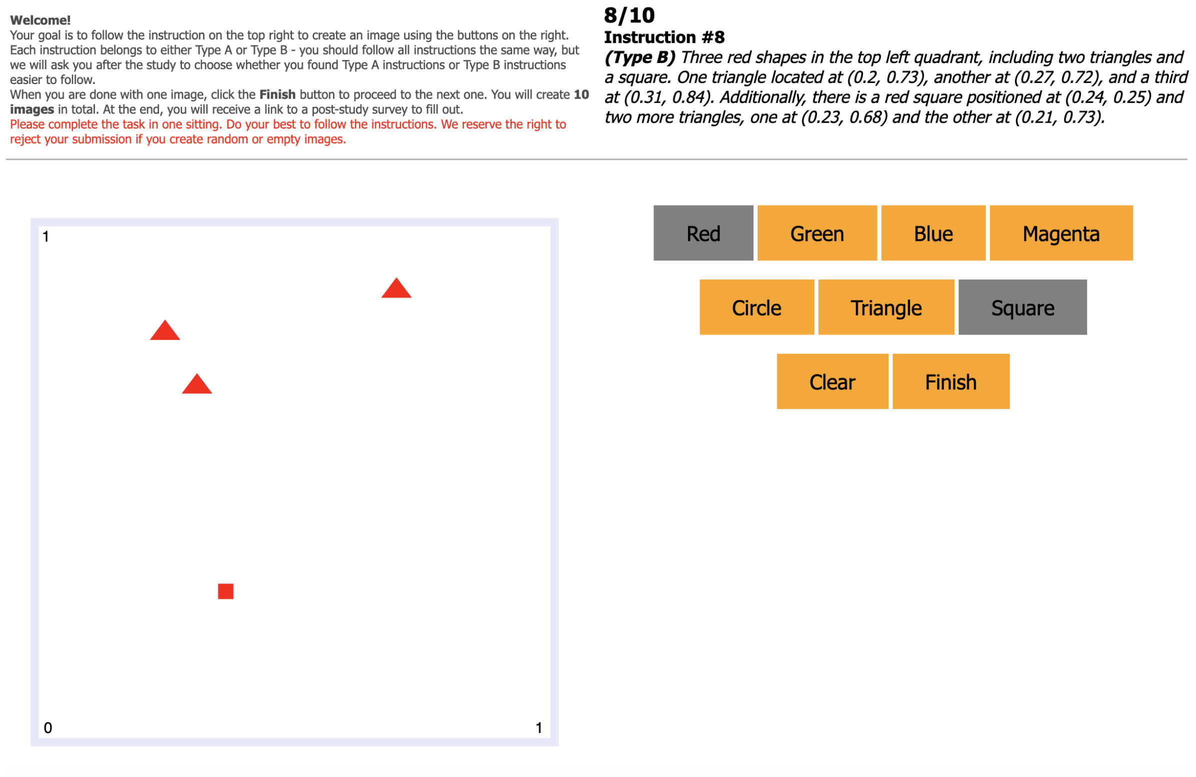


Figure 10: Interface used in the human subject study for the MAZE task. Cells not visited by the participants are hidden in gray.



Welcome!
Your goal is to follow the instruction on the top right to create an image using the buttons on the right. Each instruction belongs to either Type A or Type B - you should follow all instructions the same way, but we will ask you after the study to choose whether you found Type A instructions or Type B instructions easier to follow.
When you are done with one image, click the **Finish** button to proceed to the next one. You will create **10 images** in total. At the end, you will receive a link to a post-study survey to fill out.
Please complete the task in one sitting. Do your best to follow the instructions. We reserve the right to reject your submission if you create random or empty images.

8/10
Instruction #8
(Type B) *Three red shapes in the top left quadrant, including two triangles and a square. One triangle located at (0.2, 0.73), another at (0.27, 0.72), and a third at (0.31, 0.84). Additionally, there is a red square positioned at (0.24, 0.25) and two more triangles, one at (0.23, 0.68) and the other at (0.21, 0.73).*

Figure 11: Interface used in the human subject study for the BUILDER task.

provided by Agent 1 to Agent 2, and REWARD is the reward Agent 2 receives when trying to re-create the image (higher is better). The examples are separated into HIGH REWARD and LOW REWARD examples. + `[samples]`+

Based on the examples above, output a list of 2 RULES for Agent 1 to follow when generating INSTRUCTION in order to receive HIGH REWARD, instead of LOW REWARD. Write the rules after the prefix RULES:

4. BIRDS: The top row of three images have the following HIGH REWARD descriptions:+`high reward samples`+The bottom row of three images have the following LOW REWARD descriptions:+`low reward samples`+Provide a rule I should follow in order to provide image descriptions with HIGH REWARD, not LOW REWARD. Provide the rule after the prefix RULE:

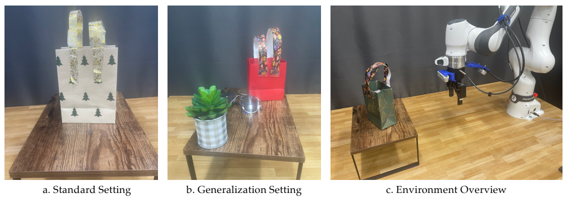


Figure 12: Task environment for Grasp, showing new location and distractors for the Generalization setting.

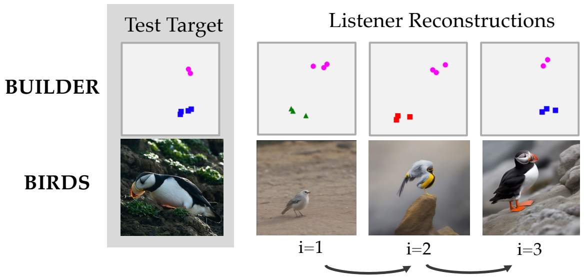


Figure 13: Both the BUILDER and BIRDS tasks consist of speaker and listener agents. At test time, the speaker needs to provide a language description to the listener that helps them recreate the image accurately. For both tasks, PLLB helps improve listener accuracy over time.

5. Grasp: The top image shows a grasp keypoint with HIGH REWARD. The bottom image shows a grasp keypoint with LOW REWARD. Based on these images, provide a rule the robot should follow in order to select a grasp keypoint that results in HIGH REWARD, not LOW REWARD. Provide the rule after the prefix RULE:

We next provide the full prompts used in `update` for each task.

1. SaySelect: [$\mathcal{L}$]+Agent 1 selected `[observation]`. So I should select
2. Maze: You are an agent solving a maze following a provided RULE. You will be given a list of PREVIOUS ACTIONS and the CURRENT OBSERVATION. Follow the RULE to select your NEXT ACTION (East, West, South, North):

   RULE: + [$\mathcal{L}$]+ PREVIOUS ACTIONS: + [$\tau_{1\ldots t-1}$] + CURRENT OBSERVATION: + `[observation]` +

   What is the NEXT ACTION you should take? Output one of (East, West, South, North) after the prefix NEXT ACTION:.
3. Builder: You will be given a DESCRIPTION of an image. Your goal is to use this description to provide a short INSTRUCTION to help someone else, who cannot see the image, accurately re-construct it. You will also be given a list of RULES you must follow when providing the instruction.

   DESCRIPTION: + `observation` +

   RULES: +$\mathcal{L}$ +

   Please provide a short instruction following the prefix INSTRUCTION:
4. Birds: Provide a one-sentence description of this image, using the following RULES:+$\mathcal{L}$
5. Grasp: Provide a keypoint in the image where the robot should grasp the object, following the RULE: $\mathcal{L}$.

B.8 Examples of Generated Rules for All Environments

See Tables 1, 2, 3, 4, and 5.

Setting	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_5$
Standard	I should follow the strategy of choosing action 4 whenever the observation is 2, 3, 4, or 5.	I should follow the strategy of choosing actions based on the observation provided by AGENT 1. If the observation is 1, I should choose action 1. If the observation is 2, I should choose action 3. If the observation is 3, I should choose action 3. If the observation is 4, I should choose action 4. If the observation is 5, I should choose action 5.	I should follow the strategy of choosing the same action as AGENT 1 for observations 1, 2, 3, 4, and 5.
Fixed Speaker	I should choose action 1 when observation is 1 or 2 or 4 or 5. I should choose action 2 when observation is 3.	I should choose action 1 when observation is 1, 2, 3 or 5. I should choose action 2 when observation is 4.	I should choose action 1 when observation is 1. I should choose action 4 when observation is 2. I should choose action 5 when observation is 3. I should choose action 2 when observation is 4. I should choose action 3 when observation is 5.

Table 1: Example $\mathcal{L}$ rules generated for the SAYSELECT environment, for the Standard setting (both Listener and Speaker agents are RL agents trained from random initialization) and a Fixed Speaker agent.

B.9 Effect of Model on Generated Rules

We additionally compare the effect of different language models, in particularly those of different sizes, on our results. We did not observe a significantly strong quantitative difference in performance when `gen_rule` is instantiated with models of different sizes (e.g. llama-2-70b-chat vs. llama-2-13b-chat). However, we did notice interesting qualitative differences across samples that are likely due to the additional fine-tuning step using reinforcmenet learning from human feedback (RLHF, see `https://llama.meta.com/llama2/` for more information). We describe these differences per environment below.

1. SAYSELECT: Because of the simplicity of this environment, it is possible for a rule $\mathcal{L}$ to summarize the full optimal human-interpretable policy as a sequence of if-statements (e.g. *"If the current state is '1', I should take action 1."*) - we observed that smaller language models (e.g. llama-2-7b-chat and llama-2-13b-chat) always did this, while larger models (e.g. llama-2-70b-chat and mistral-8x-7b-instruct) were better able to generalize and use more efficient language, such as *"I should take the same action as the observation"*. The smaller models also often included superfluous language, such as *"I should always take the action that leads to the highest reward"*.

2. MAZE: The majority of generated rules captured the underlying color semantics of the maze, enabling generalization. However, smaller model sizes (e.g. llama-2-7b-chat) resulted in more superfluous language (e.g. *"I should always prioritize taking actions that lead to the most recent reward, and avoid taking actions that lead to low reward."*), and the generated rules focused more on the first few actions the agent should take, which may not always generalize.

3. BUILDER: While we do not observe any model-specific differences, there exists variation across samples in the type of formatting and syntax generated rules encourage (e.g. provide coordinates *" using the format (x, y)"* vs. *" using the format "x=0.XX, y=0.YY""*), leading to agents converging to different descriptions. Furthermore, some rules encourage list formats in image descriptions (e.g. *1.Draw a green dot at (0.72, 0.21). 2. Draw a green dot at (0.73, 0.72).*) while other rules encouraged clustering of identical shapes (e.g. *Draw two green dots at (0.72, 0.21) and (0.73, 0.72).*)

Size	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_5$	$\mathcal{L}_8$
7x7	I should start by moving north, then continue moving north until I reach a wall, then turn east and move until I reach a wall, then turn south and move until I reach a wall, then turn west and move until I reach the end of the maze.	I should prioritize moving North, East, and South over moving West, as the HIGH REWARD EXAMPLES show a higher frequency of these actions. 2. I should avoid moving West unnecessarily, as it is rarely seen in the HIGH REWARD EXAMPLES and often leads to LOW REWARD EXAMPLES	I should take the EAST action when I see WHITE, and then repeat the EAST action until I see RED, then take the SOUTH action, and then repeat the EAST action until I see BLUE, then take the NORTH action, then take the EAST action until I see RED again, then take the SOUTH action, and finally take the WEST action.	I should take the EAST action when I observe WHITE, and keep taking EAST actions until I observe RED, then take the SOUTH action. After taking the SOUTH action, I should take the EAST action again until I observe BLUE, then take the NORTH action. After taking the NORTH action, I should take the EAST action until I observe RED again, then take the SOUTH action.

Table 2: Example $\mathcal{L}$ rules generated for the MAZE environment for the 7x7 maze size.

Setting	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$
Re-Initialization	1. Be specific with location details: Agent 1 should provide detailed location information for each element in the image, such as x and y coordinates. 2. Use descriptive language for elements, such as "red dot" or "green triangle".	1. Use specific coordinates when instructing Agent 2 to draw shapes. 2. Use descriptive language to specify the color and shape of each element. For example, "a green triangle" instead of "a green thing".	1. Be specific with location coordinates: provide specific coordinates for the location of each shape, using the format x=0.XX, y=0.YY. 2. Use descriptive shape names: Instead of using generic terms like "dot" or "square," use more descriptive names that indicate the shape's color and size, such as "green triangle" or "red square."
Continual Training	1. Be specific and detailed in your instructions. High reward examples have specific coordinates and shapes, while low reward examples have more general descriptions. 2. Use a consistent format for your instructions. High reward examples have a consistent format for listing coordinates and shapes, while low reward examples have a more free-form format.	1. Provide explicit coordinates for each element in the image, using the format (x, y). 2. Use specific colors when referring to elements in the image, such as "red", "green", or "blue". Avoid using vague terms like "colored" or "shaded".	1. Use a consistent format for describing shapes, such as always listing the x-coordinate first, followed by the y-coordinate. For example, instead of "one green square at the point x=0.53, y=0.24", use "one green square at (0.53, 0.24)". 2. Avoid using vague terms like "various shades of green". Instead, use specific colors, such as "green" or "blue". Additionally, use specific shapes, such as "square" or "triangle", rather than vague terms like "rectangle".

Table 3: Example $\mathcal{L}$ rules generated for the BUILDER environment.

4. BIRDS: Rules demonstrated more reward-specificity (i.e. specific to background, color, or species rewards) when generated with larger VLMs (e.g. llava-13b) versus smaller models (e.g. llava-v1.6-vicuna-7b), with the latter primarily proposing rules that encouraged more detailed descriptions (e.g. *"Avoid using vague or general terms"*).

5. GRASP: We found that most VLMs other than molmo-7B performed poorly at recognizing keypoints, and instead relied heavily on task information. For example, one rule generated by llava-13b was *"A robot should grasp a keypoint that is visible and not obstructed by other objects."* We believe the superior performance of the molmo series of models are due to specifically training on the PixMo dataset with 2D-points (Allen Institute for AI).

Reward	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$
color	Describe the bird's color, species, and any distinctive markings or patterns.	Describe the bird's coloration accurately.	Describe the bird's coloration accurately.
background	Include details about the bird's surroundings, such as the type of branch or post it is on, and any additional elements in the background.	Include the bird's action (perched, flying, standing) and its location (on a branch, railing, pole, etc.)	Describe the bird's action (flying, perching, standing) and the environment it is in (sky, tree, water).
species	Describe the bird's color, markings, and any distinctive features.	Describe the subject's unique features, such as coloration, beak shape, or other distinguishing characteristics.	Include specific details about the bird's appearance, such as the color of its feathers, beak, or eyes, and any distinctive markings or patterns.

Table 4: Example $\mathcal{L}$ rules generated for the BIRDS environment demonstrate reward-specificity over time.

Reward	$\mathcal{L}_{10}$	$\mathcal{L}_{30}$	$\mathcal{L}_{50}$
Shoulder Torque	Select keypoints that are away from sharp edges or corners of the object to avoid potential damage and improve stability during grasping.	Prioritize grasp keypoints that are closer to the center of the object for more stable and precise pickup.	Prioritize grasping at the middle of the object's surface rather than the top edges or handles to ensure a stable grip.

Table 5: Example $\mathcal{L}$ rules generated for the GRASP environment.