SayCan: Grounding Language in Robotic Affordances

Peizhen Li[1]

[1]Faculty of Science and Engineering
Macquarie University

11 Sep 2023

MACQUARIE University

Outline

1 SayCan: Grounding Language in Robotic Affordances
- Introduction
- Preliminaries
- Method
- Implementing SayCan in a Robotic System

2 Experimental Evaluation
- Experimental Setup & Evaluation Metrics
- Results
- Case studies of New Capabilities

3 Conclusions, Limitations & Future Work
- Conclusions and Limitations
- Future Work

SayCan

Do As I Can, Not As I Say
Grounding Language in Robotic Affordances

- Uses LLM capabilities for robot agents without additional model training
- Grounds LLMs (Say) through affordance functions (Can)
- Generates feasible plans for robots
- Can be integrated with chain-of-thought prompting to handle tasks that require reasoning

SayCan

Figure: LLMs have not **interacted with their environment** and **observed the outcome** of their responses, and thus are not grounded in the world. SayCan grounds LLMs via value functions.

Large Language Models

- Language Models:
  $p(W) = \prod_{j=0}^{n} p(w_j | w_{<j}), W = \{w_0, w_1, \cdots, w_n\}$
- Large Language Models (LLMs): Transformers, BERT, GPT-3, LAMDA, and PaLM etc.

Value Functions and RL

Goal: Accurately predict whether a skill is feasible at a current state.

- A Markov Decision Process (MDP): $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$
- State-transition probability function: $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}_+$
- Reward function: $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
- Action value function (Q-function): $Q^\pi(s, a) = \mathbb{E}_{a \sim \pi(a|s)} \sum_t R(s_t, a_t)$
- Temporal-difference (TD) based methods

SayCan: Do As I Can, Not As I Say

Problem Statement

- Given
  - a set of skills $\prod$
  - each skill $\pi \in \prod$
  - comes with a language description (textual label) ℓ_π
- An affordance function $p(c_\pi | s, \ell_\pi)$
- The system receives a natural language instruction i
- The LLM provides us with $p(\ell_\pi | i)$
- Probability of actually completing the instruction $p(c_i | i, s, \ell_\pi) \propto p(c_\pi | s, \ell_\pi) p(\ell_\pi | i)$

SayCan: Do As I Can, Not As I Say

Connecting Large Language Models to Robots

- Break down the high-level instruction into available low-level skills
  - Prompt engineering
  - Constrained responses: scoring language models
  - Iteratively select a skill and appending it to the instruction.

$$\pi = \arg\max_{\pi \in \prod} p(c_\pi | s, \ell_\pi) p(\ell_\pi | i) \tag{1}$$

SayCan: Do As I Can, Not As I Say

- Ground large language models through value functions

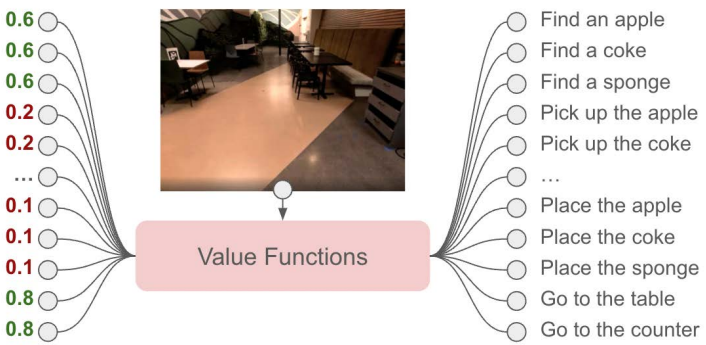
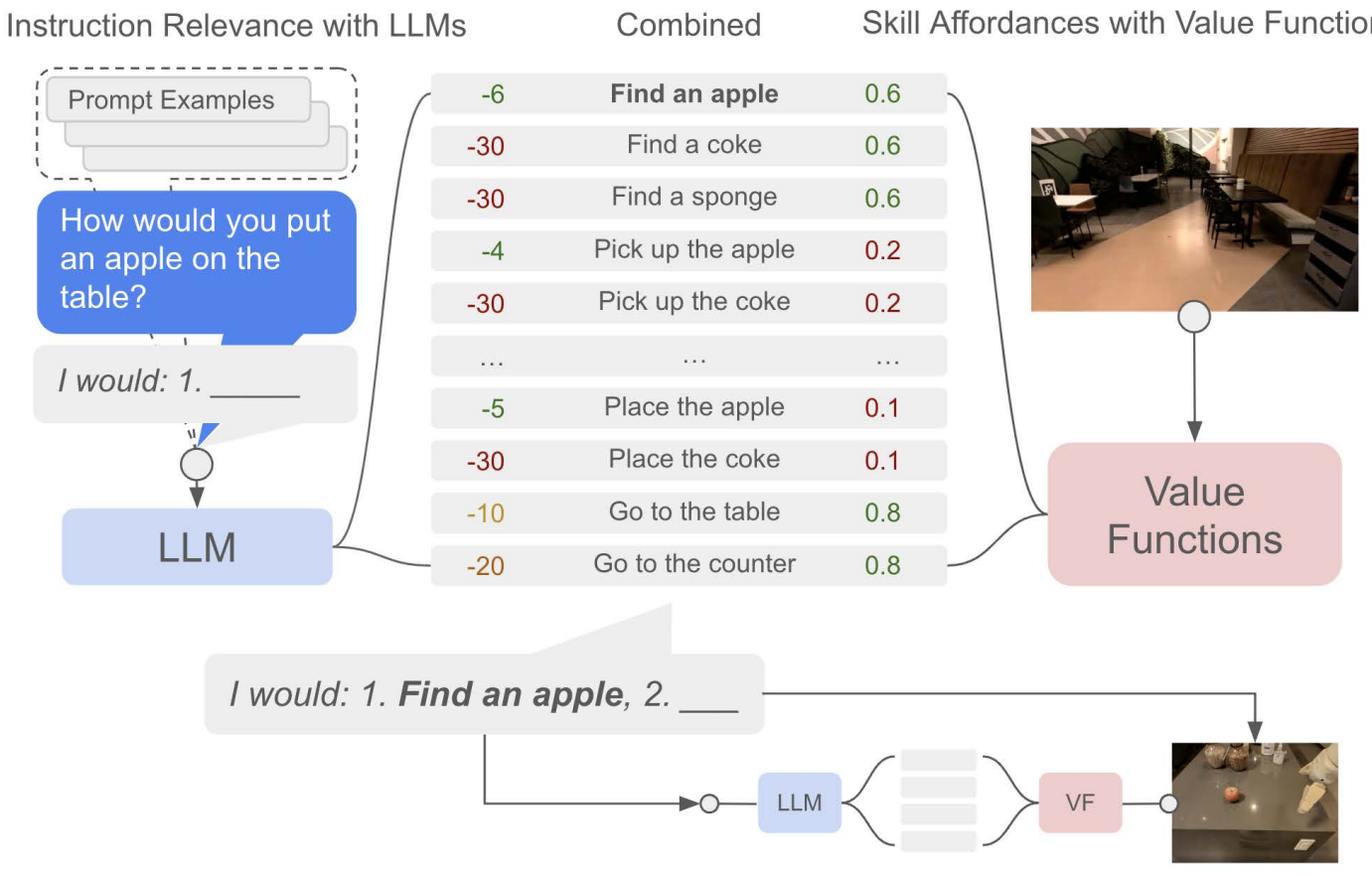


Figure: Value function space

Figure: Visualization

SayCan: Do As I Can, Not As I Say

Interpretability:
structure the planning as a dialog between a user and a robot



SayCan: Do As I Can, Not As I Say

Algorithm 1: SayCan

Given: A high-level instruction i, state s_0, skill set $\prod$ and $\ell_{\prod}$

$n = 0, \pi = \emptyset$
while $\ell_{\pi_{n-1}} \neq$ *"done"* **do**
  $\mathcal{C} = \emptyset$
  for $\pi \in \prod$ *and* $\ell_\pi \in \ell_{\prod}$ **do**
    $p_\pi^{LLM} = p(\ell_\pi | i, \ell_{\pi_{n-1}}, \cdots, \ell_{\pi_0})$
    $p_\pi^{affordance} = p(c_\pi | s_n, \ell_\pi)$
    $p_\pi^{combined} = p_\pi^{affordance} p_\pi^{LLM}$
    $\mathcal{C} = \mathcal{C} \cup p_\pi^{combined}$
  end
  $\pi_n = \arg\max_{\pi \in \prod} \mathcal{C}$, Execute $\pi_n(s_n)$, update state s_{n+1},
  $n = n + 1$
end

Language-Conditioned Robotic Control Policies

- Obtain policies and value functions for given skills

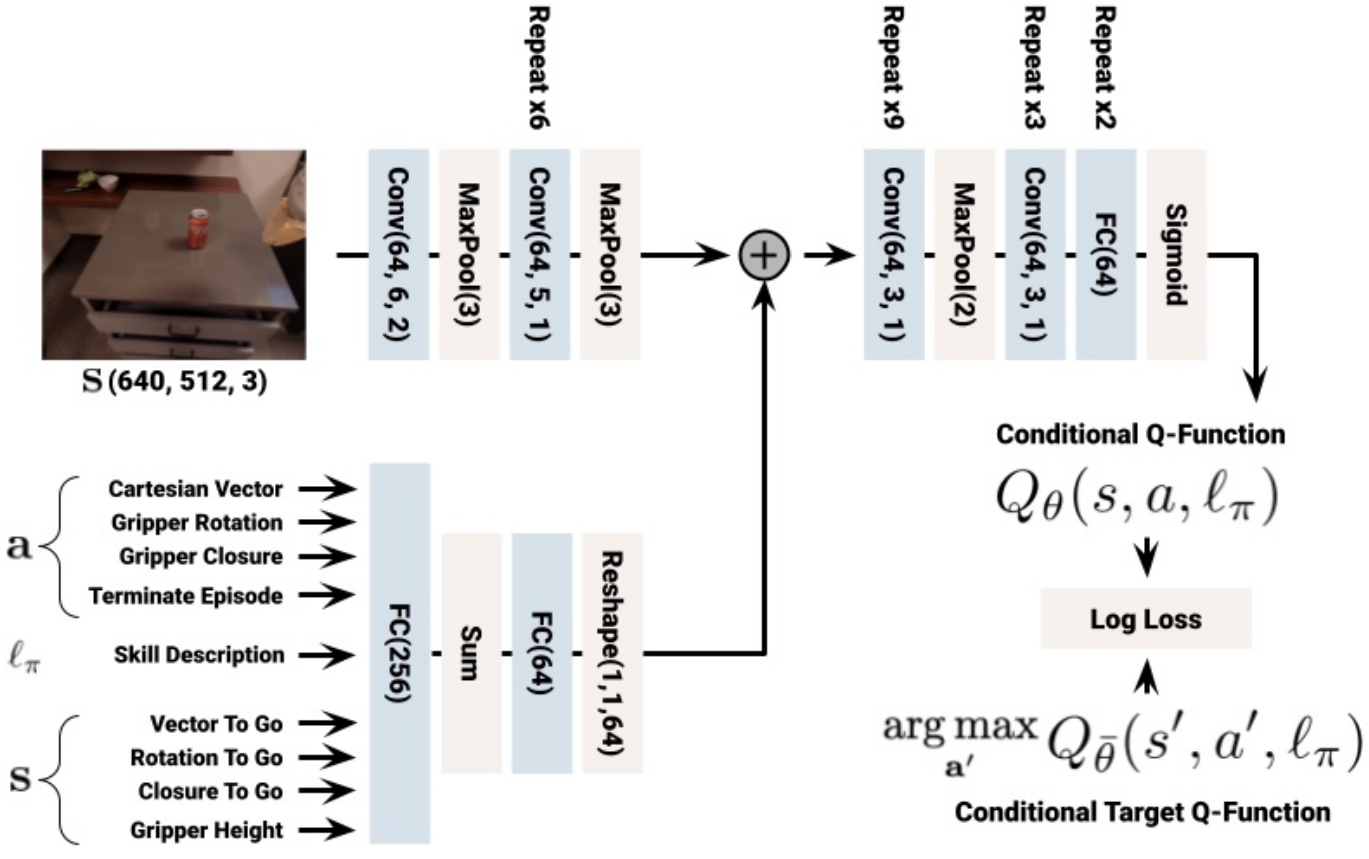


Figure: Nework architecture in RL policy

Training the Low-Level Skills

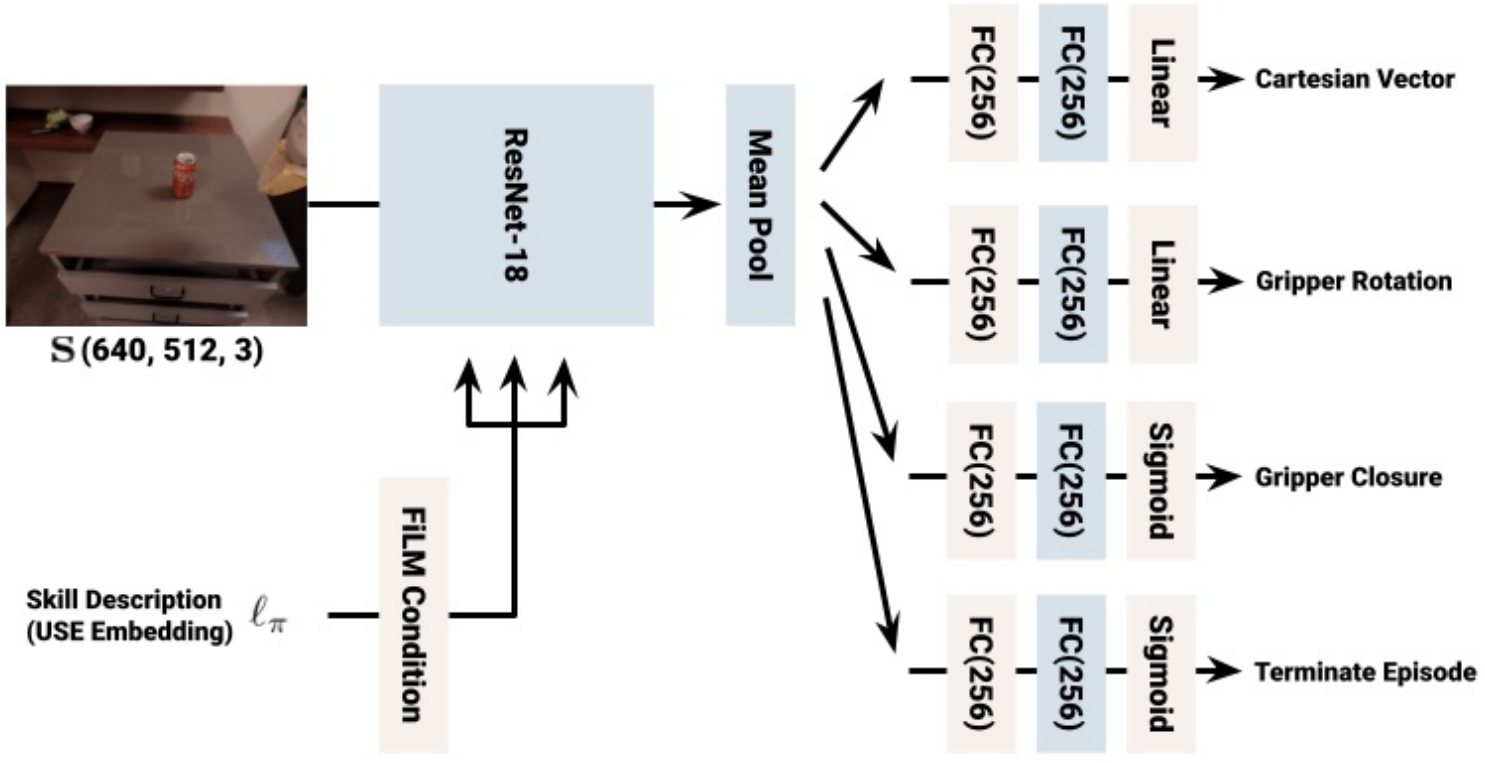


Figure: Nework architecture in BC policy

Experimental Setup

The LLM used is 540B PaLM

Figure: Office kitchen

Figure: 15 objects

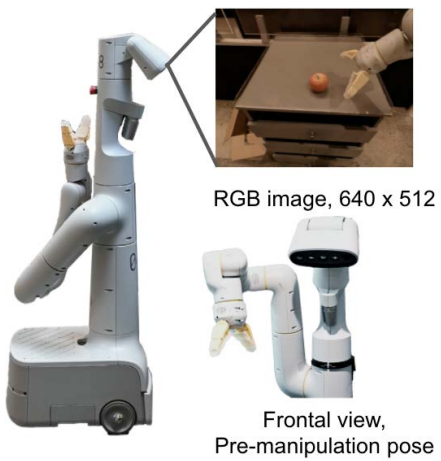


Figure: Mobile manipulator

Instructions and Metrics

- Test across 101 instructions from 7 instruction family

Instruction Family	Num	Explanation	Example Instruction
NL Single Primitive	15	NL queries for a single primitive	Let go of the coke can
NL Nouns	15	NL queries focused on abstract nouns	Bring me a fruit
NL Verbs	15	NL queries focused on abstract verbs	Restock the rice chips on the far counter
Structured Language	15	Structured language queries, mirror NL Verbs	Move the rice chips to the far counter
Embodiment	11	Queries to test SayCan's understanding of the current state of the environment and robot	Put the coke on the counter. (starting from different completion stages
Crowd-Sourced	15	Queries in unstructured formats	My favorite drinks is redbull, bring one
Long-Horizon	15	Long-horizon queries that require many steps of reasoning	I spilled my coke on the table, throw it away and bring me something to clean

- Metrics
  - plan success rate
  - execution success rate

Results

- Performance of PaLM-SayCan across 101 tasks

		Mock Kitchen		Kitchen		No Affordance		No LLM	
		PaLM-SayCan	**PaLM-SayCan**	**PaLM-SayCan**	**PaLM-SayCan**	**No VF**	**Gen.**	**BC NL**	**BC USE**
Family	**Num**	Plan	Execute	Plan	Execute	Plan	Plan	Execute	Execute
NL Single	15	100%	100%	93%	87%	73%	87%	0%	60%
NL Nouns	15	67%	47%	60%	40%	53%	53%	0%	0%
NL Verbs	15	100%	93%	93%	73%	87%	93%	0%	0%
Structured	15	93%	87%	93%	47%	93%	100%	0%	0%
Embodiment	11	64%	55%	64%	55%	18%	36%	0%	0%
Crowd Sourced	15	87%	87%	73%	60%	67%	80%	0%	0%
Long-Horizon	15	73%	47%	73%	47%	67%	60%	0%	0%
Total	101	84%	74%	81%	60%	67%	74%	0%	9%

- Ablating the language model

		PaLM-SayCan		**FLAN-SayCan**	
Family	**Num**	Plan	Execute	Plan	Execute
NL Single	15	100%	100%	67%	67%
NL Nouns	15	67%	47%	60%	53%
NL Verbs	15	100%	93%	80%	67%
Structured	15	93%	87%	100%	87%
Embodiment	11	64%	55%	64%	55%
Crowd Sourced	15	87%	87%	73%	67%
Long-Horizon	15	73%	47%	47%	33%
Total	101	84%	74%	70%	61%

Adding Skills

- Integrating new skills
  - Adding new skills as options for the LLM
  - Providing accompanying value functions
  - Adding and example in the prompt with that skill

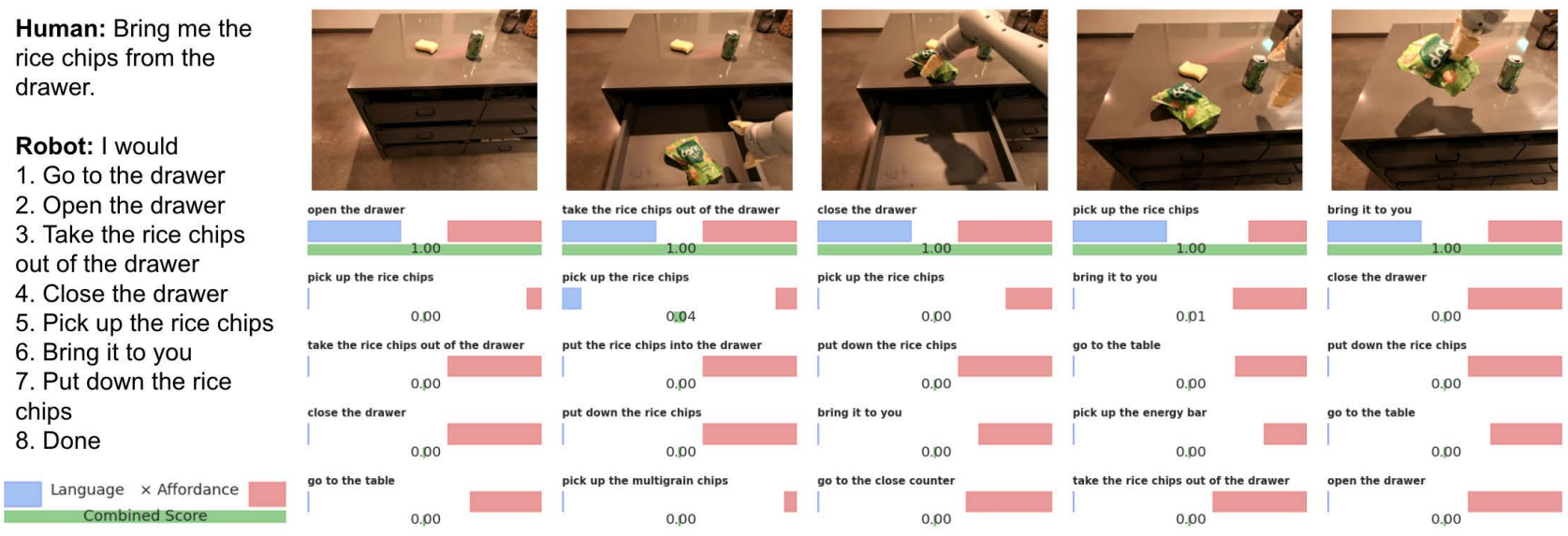


Figure: An example of drawer manipulation (long-horizon planning)

Chain of Thought Reasoning

- Modify the prompt to include a part called “Explanation”

CHAIN-OF-THOUGHT PLANNING ROLLOUT WITH PALM-SAYCAN.

Human: Can you bring a fruit-flavored drink without caffeine?

Explanation: The user has asked for a drink that is fruit-flavored and does not have caffeine, I will bring the lime soda.

Robot: 1. find a lime soda, 2. pick up the lime soda, 3. bring it to you, 4. put down the lime soda, 5. done

Human: Bring me the spicy chips.

Explanation: The user has asked for chips that are spicy, I will bring the Jalapeno chips.

Robot: 1. find the jalapeno chips, 2. pick up the jalapeno chips, 3. bring it to you, 4. put down the jalapeno chips, 5.done

Human: The snack is too small, bring me something more filling.

Explanation: The user has asked for a snack that is more filling, I will bring the multigrain chips.

Robot: 1. find the multigrain chips, 2. pick up the multigrain chips, 3. bring it to you, 4. put down the multigrain chips, 5. done

---

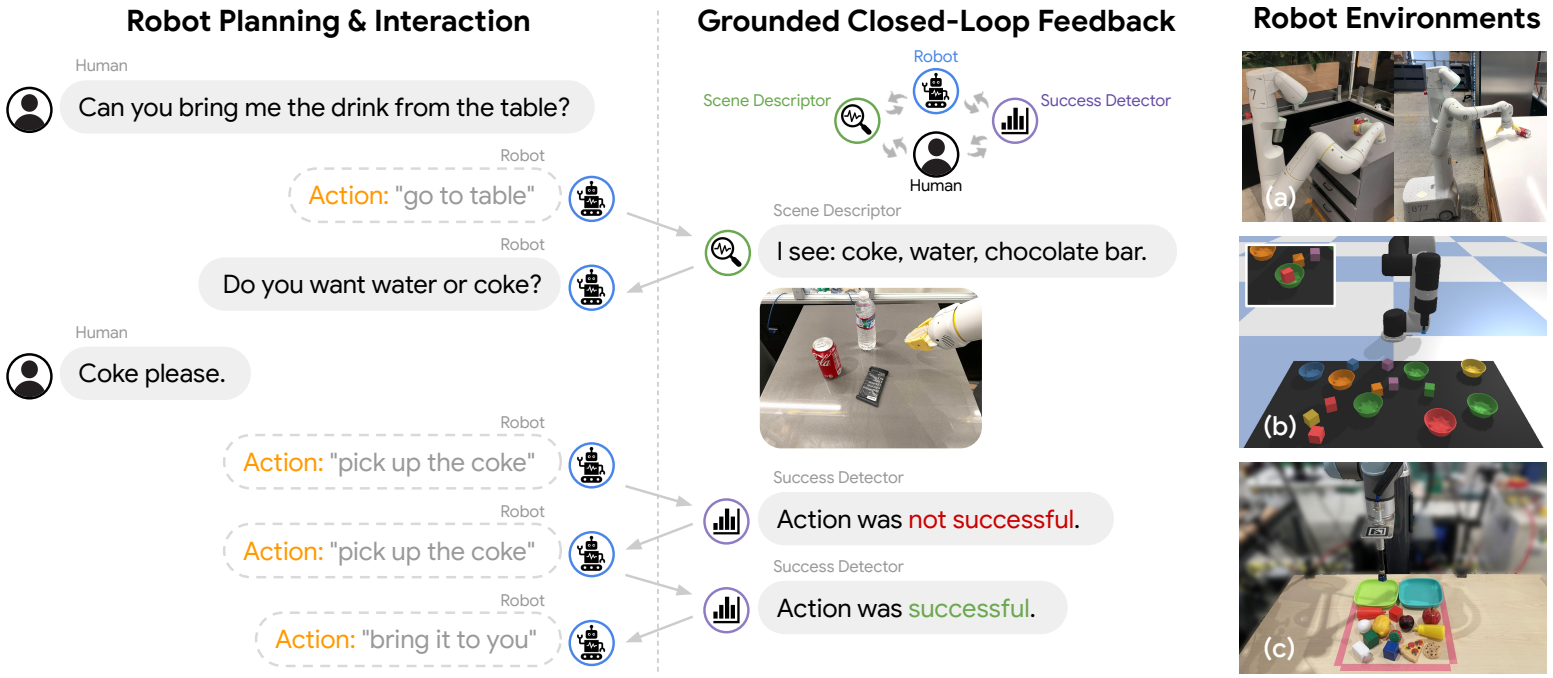
Multilingual Queries

- The underlying LM has been trained on multilingual corpora

Instruction	**Plan rate**
bring me a can of coke	1.0
throw away the coke can	1.0
I spilled my coke, can you bring me something to help clean	1.0
拿一罐可乐给我	1.0
扔掉可乐罐	1.0
我的可乐洒了，你能给我拿点东西来帮忙打扫吗	1.0
apporte moi une canette de coca	1.0
jeter la canette de coca	1.0
J'ai renversé mon coca, peux-tu m'apporter quelque chose pour m'aider à nettoyer	0.0
tráeme una lata de coca cola	1.0
tirar la lata de coca cola	1.0
Derramé mi coca cola, ¿puedes traerme algo para ayudar a limpiar	1.0

Closed-Loop Planning: Inner Monologue

- Leveraging various sources of environment feedbacks
- Continually injecting information into the LLM planning prompts



Open Source Environment

- Single step selection: affordance scoring, LLM scoring

```
termination_string = "done()"
query = "To pick the blue block and put it on the red block, I should:\n"

options = make_options(PICK_TARGETS, PLACE_TARGETS, termination_string=termination_string)
llm_scores, _ = gpt3_scoring(query, options, verbose=True, engine=ENGINE)

affordance_scores = affordance_scoring(options, found_objects, block_name="box", bowl_name="circle",
                                       verbose=False, termination_string=termination_string)

combined_scores = {option: np.exp(llm_scores[option]) * affordance_scores[option] for option in options}
combined_scores = normalize_scores(combined_scores)
selected_task = max(combined_scores, key=combined_scores.get)
```

Open Source Environment

- OpenAI completions API

```
def gpt3_call(engine="text-ada-001", prompt="", max_tokens=128, temperature=0,
              logprobs=1, echo=False):
  full_query = ""
  for p in prompt:
    full_query += p
  id = tuple((engine, full_query, max_tokens, temperature, logprobs, echo))
  if id in LLM_CACHE.keys():
    response = LLM_CACHE[id]
  else:
    response = openai.Completion.create(engine=engine,
                                        prompt=prompt,
                                        max_tokens=max_tokens,
                                        temperature=temperature,
                                        logprobs=logprobs,
                                        echo=echo)
    LLM_CACHE[id] = response
  return response
```

Open Source Environment

- Iterative process of planning

```
affordance_scores = affordance_scoring(options, found_objects, block_name="box", bowl_name="circle", verbose=False)
num_tasks = 0
selected_task = ""
steps_text = []
while not selected_task == termination_string:
  num_tasks += 1
  if num_tasks > max_tasks:
    break
  # completions api
  llm_scores, _ = gpt3_scoring(gpt3_prompt, options, verbose=True, engine=ENGINE, print_tokens=False)
  combined_scores = {option: np.exp(llm_scores[option]) * affordance_scores[option] for option in options}
  combined_scores = normalize_scores(combined_scores)
  selected_task = max(combined_scores, key=combined_scores.get)
  steps_text.append(selected_task)
  print(num_tasks, "Selecting: ", selected_task)
  gpt3_prompt += selected_task + "\n"     # append selection to the prompt
```

Conclusion and Limitations

- Conclusions
  - Grounds LLMs (Say) through affordance functions (Can)
  - Generates feasible and contextually appropriate plans for robots
  - Improves a robot's performance by enhancing the underlying language model
- Limitations
  - Dependence on the training data
  - Bottleneck: the range and capabilities of the underlying skills
  - Struggle to react to situations where individual skills fail

Future Work

- Improve the language model itself by leveraging real-world robotic experience
- Other sources of grounding (instead of using value function to score affordances)
- Other ways of combining robot planning, interaction and language
- Whether natural language is the right ontology to use to program robots

Vision-Language-Action Model

Robotics Transformer 2

- Learn to map robot observations to actions
- Co-fine-tune vision-language models on both robotic trajectory data and Internet-scale vision-language tasks
- Express robotic actions as text tokens

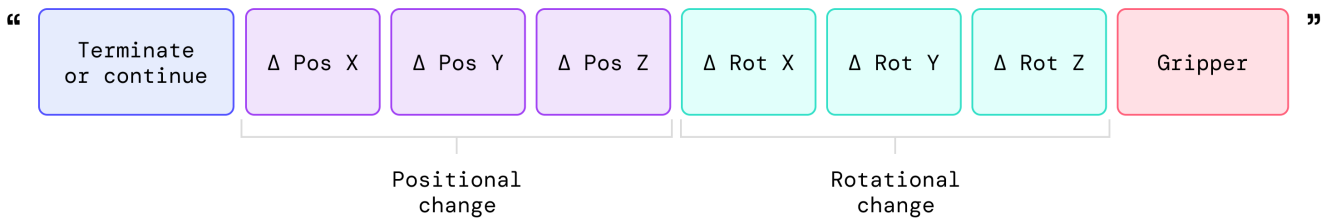
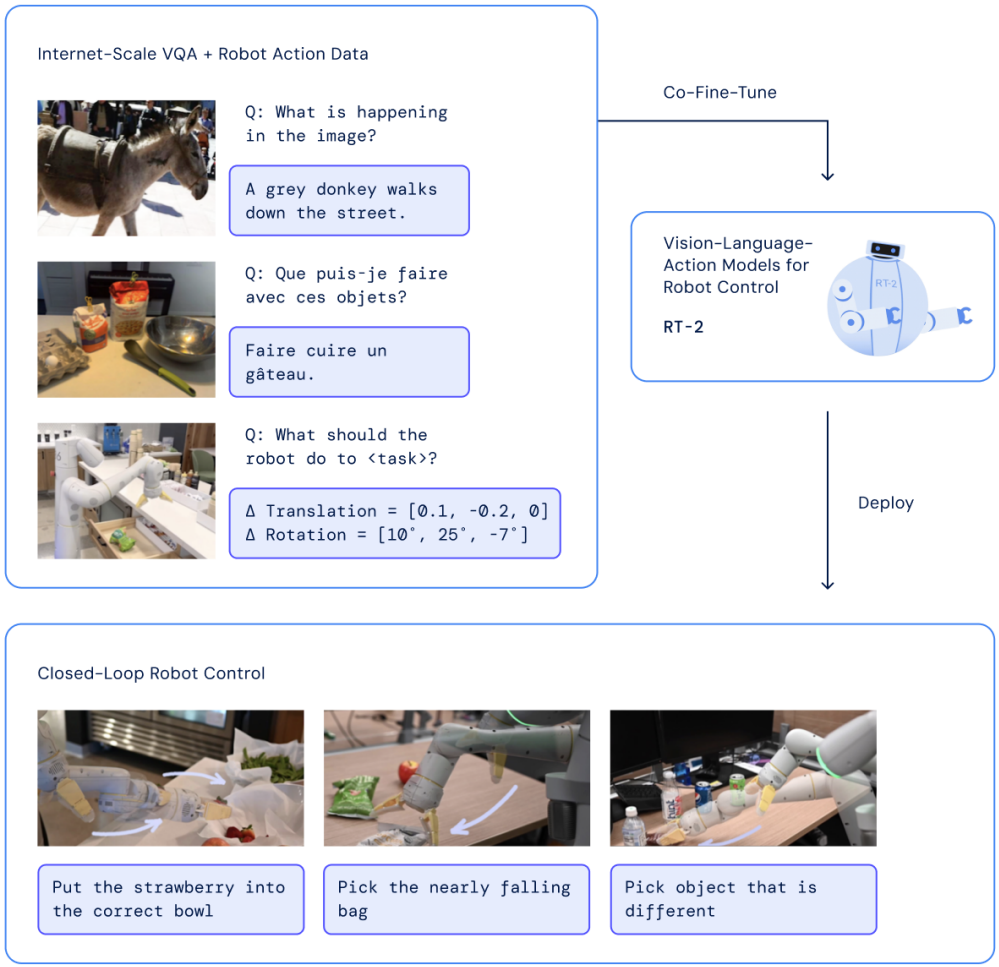


Figure: Robot action token numbers: “1 128 91 241 5 101 127 217”

Vision-Language-Action Model

Approach overview of RT-2



Limitations of RT-2

- Limitations
  - Robots do not acquire any **new physical skills** from web-scale data
  - Computation cost is high while demanding high-frequency control and real-time inference
- Future directions
  - Motion/animation capture from videos of humans (**animation retargeting**)
  - Quantization and distillation techniques

Research Direction

Robot Learning

Goal: expanding robots' **perception** and physical **interaction** capabilities

- **Multi-model perception:** harnessing vision, touch, audio, and language for fine-grained and effective manipulation
- **Embodied intelligence:** focusing on long-horizon planing, generalization to diverse environments, and sim-to-real transfer
- **Intuitive physics:** learning structured world models for robotic manipulation of objects with diverse physical properties

Thank you very much!
Q&A