Мультимодальные и поведенческие модели

Pretraining Objectives and Auxiliary Tasks

MIPT VLA 2026 | 12 октября 2026

Алексей Староверов



МФТИ

Center for Cognitive Modeling

План лекции

Embodied Reasoning

Post-Training Policies without Post-Training

Steerable Vision-Language-Action Policies for Embodied Reasoning and Hierarchical Control

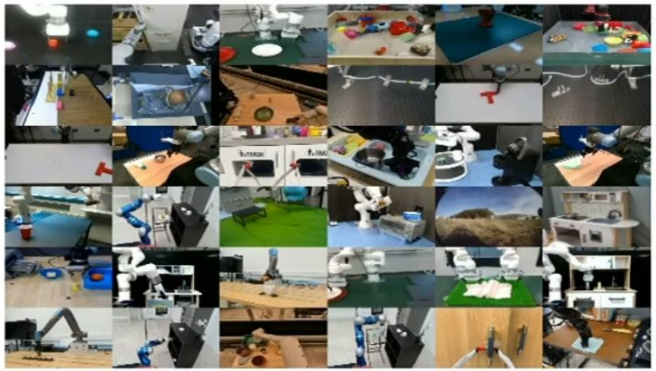
VLA Grounder: Language-Conditioning Space Optimization for Black-Box VLA Models

Is Scaling Robot Data All We Need?

No…

What is missing?

- Training LLMs on all of Internet not enough
- LLMs train specifically for reasoning



Robots capable of **many tasks**

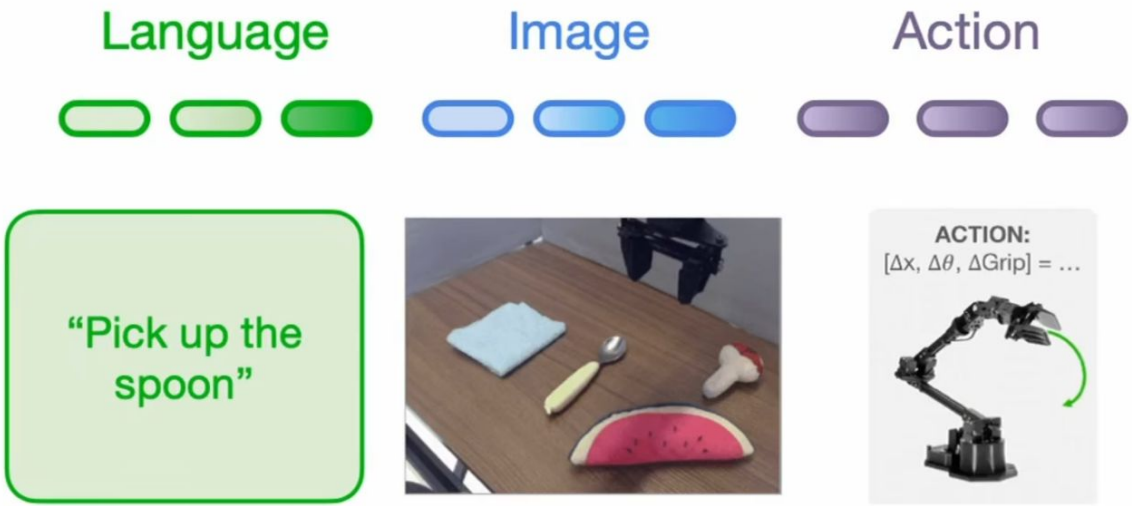
ChatGPT

deepseek

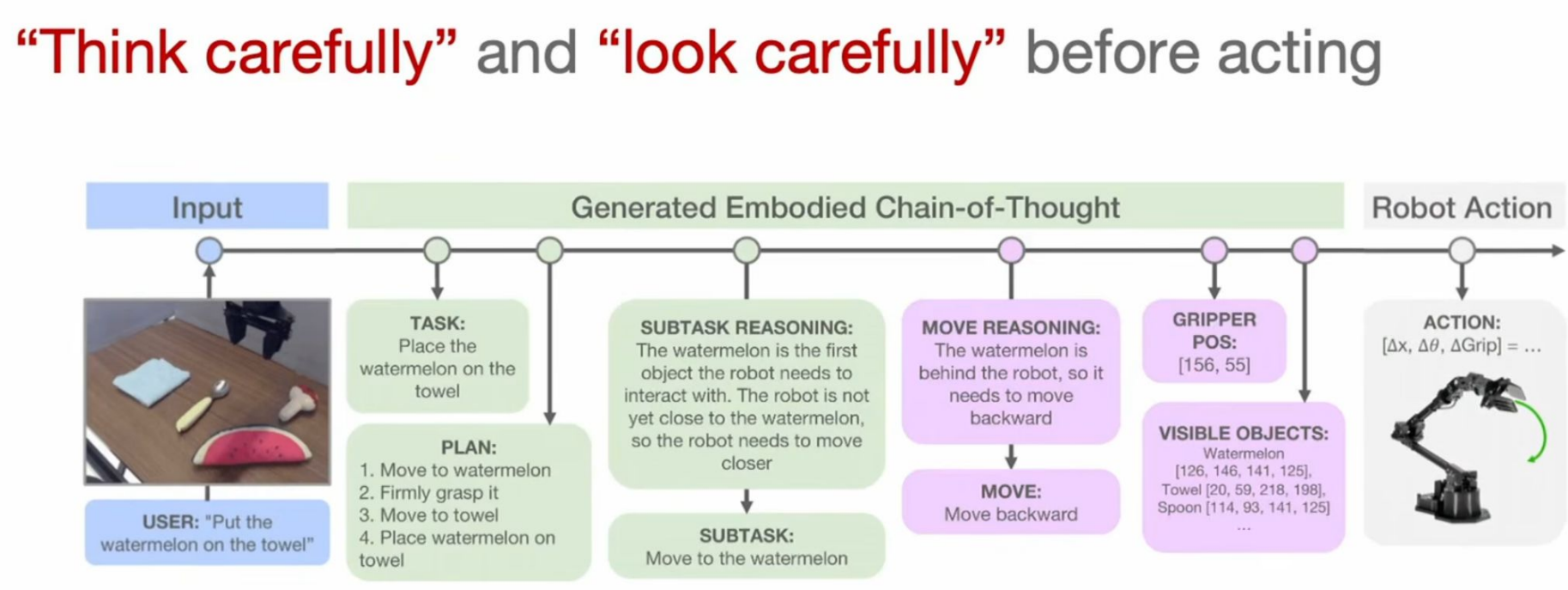
Intelligent Reasoning: Generalization, Interpretability, Interaction, Controllability

Robotics as Multimodal Sequence Modeling

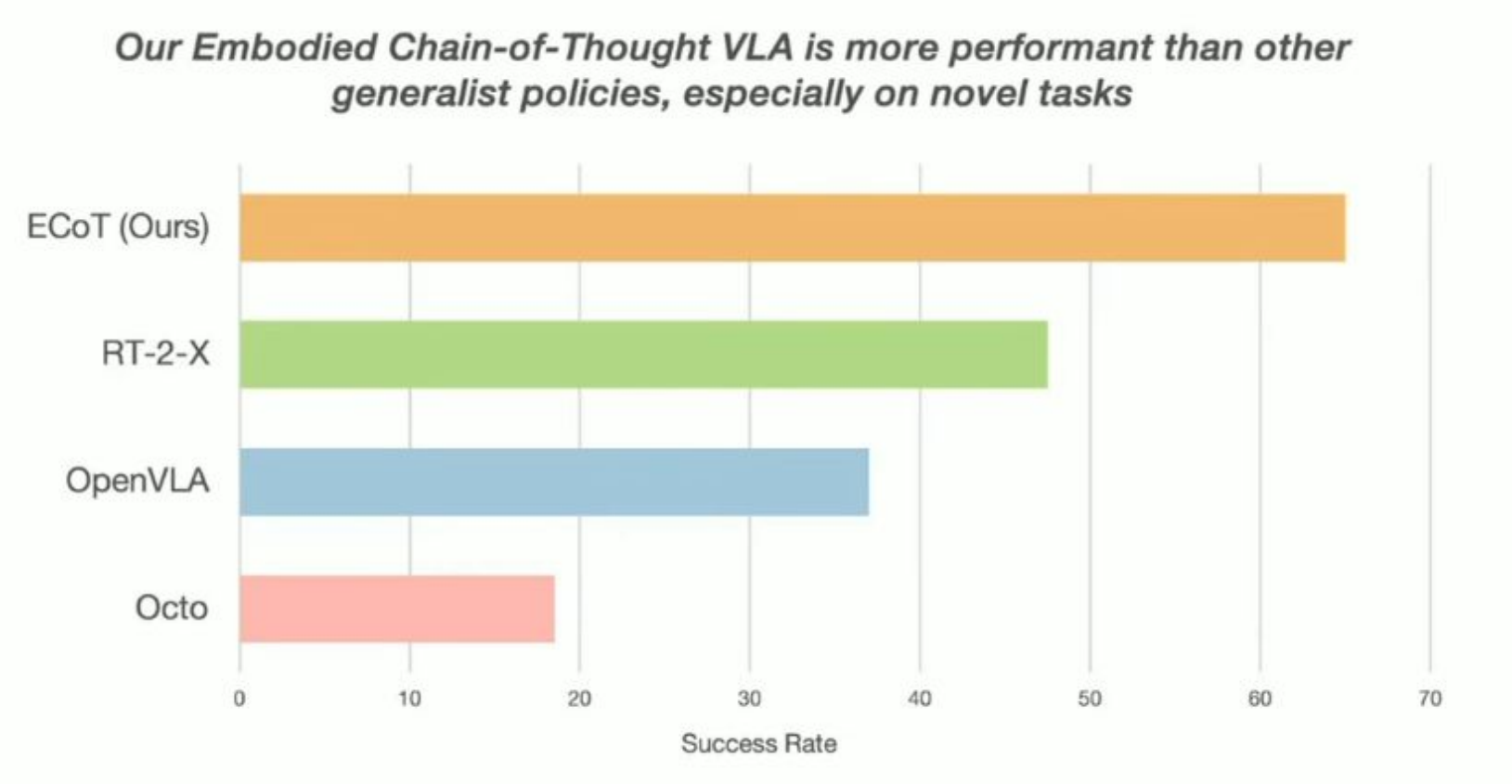
- Can we **transfer** algorithmic improvements like **reasoning**?
- Can a robot **think harder** to **act better**?



Embodied Chain-of-Thought Architecture



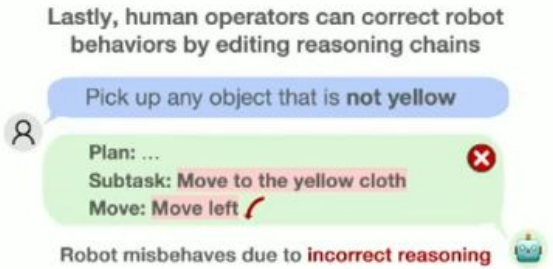
Embodied Chain-of-Thought Results



Human-Robot Interaction

- ECoT makes policies more interpretable
- Allows human to correct behaviors





Inference Cost of Reasoning

Autoregressive VLA Inference Speed
OpenVLA

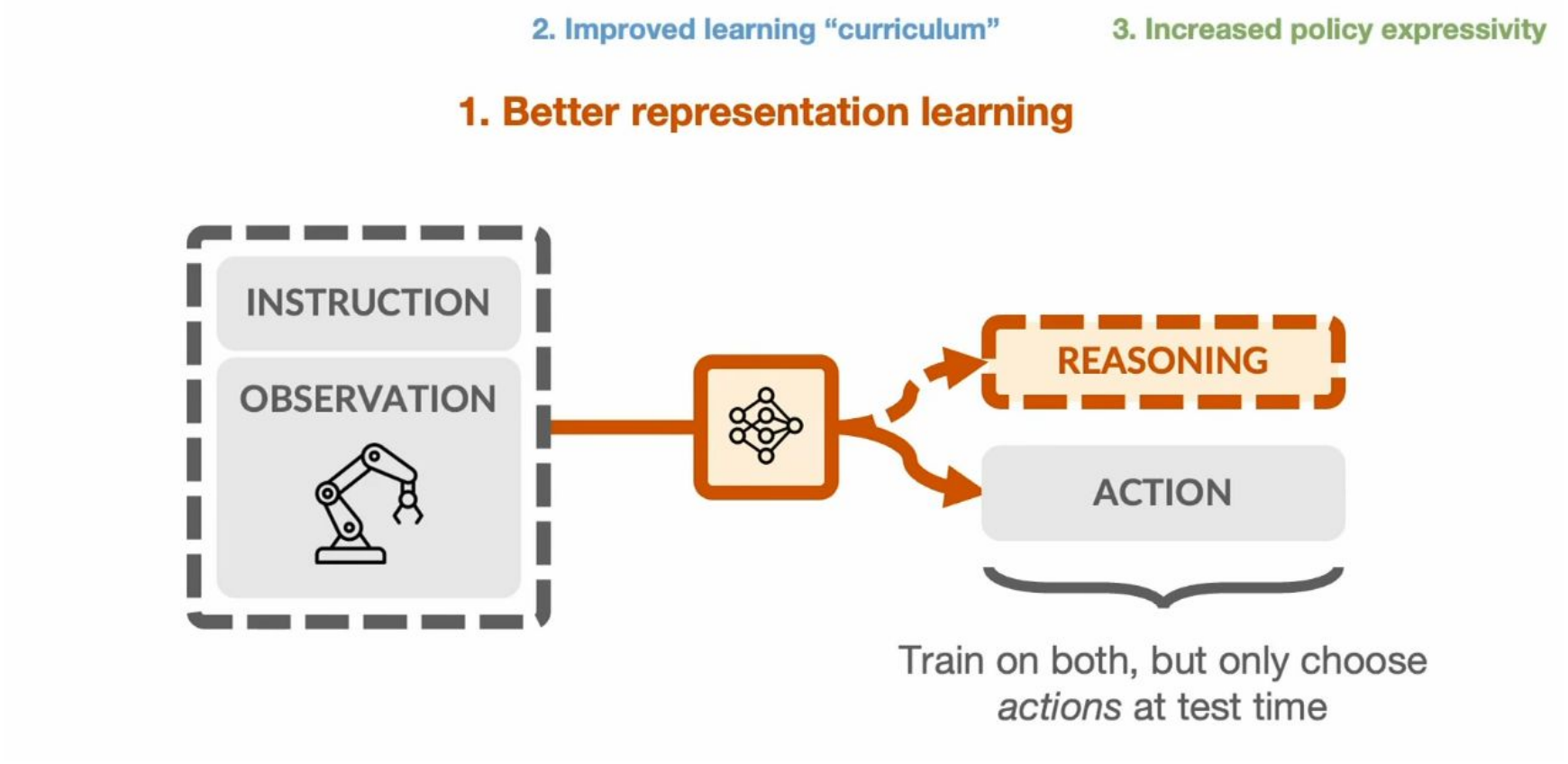
4 Actions/Second

Embodied Chain-of-Thought Reasoning Inference Speed

4 **Seconds/Action**



Why Does Embodied Reasoning Help Robot Policies?

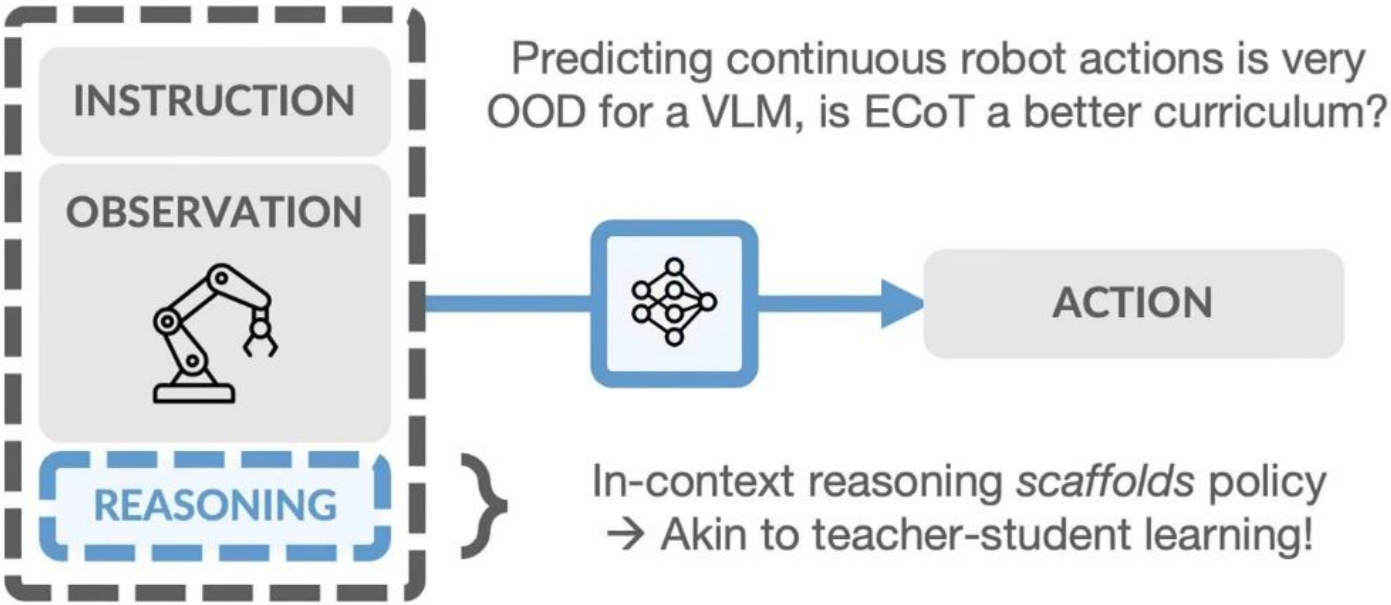


Why Does Embodied Reasoning Help Robot Policies?

1. Better representation learning

3. Increased policy expressivity

2. Improved learning "curriculum"

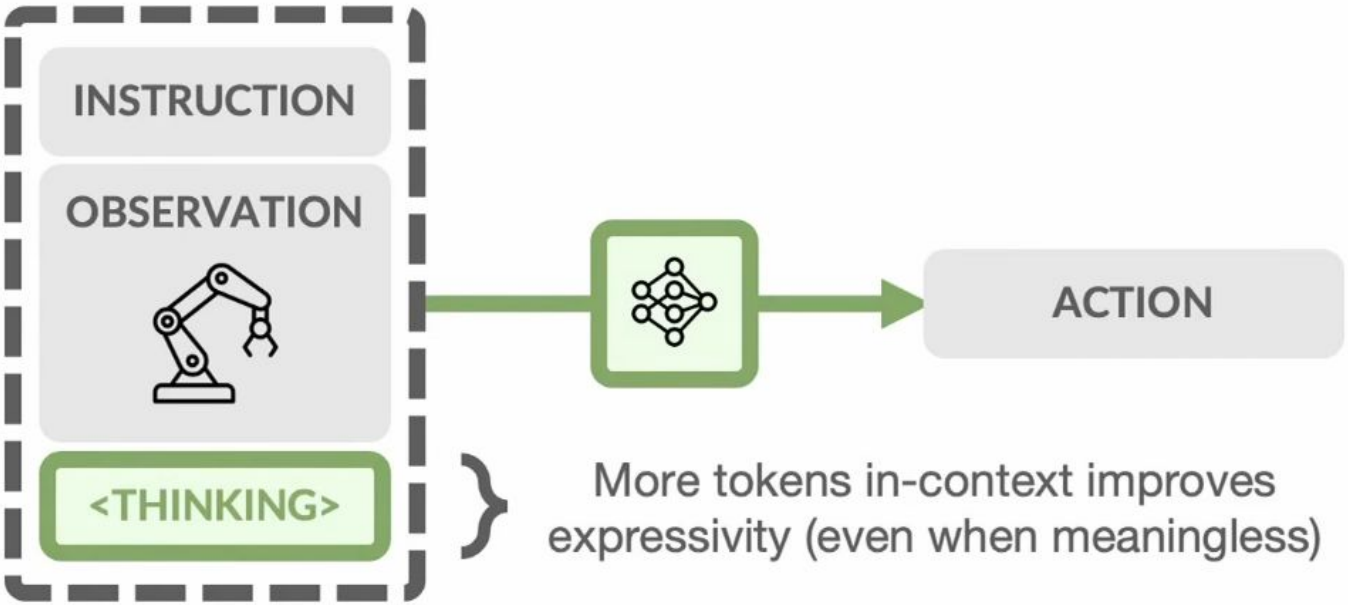


Why Does Embodied Reasoning Help Robot Policies?

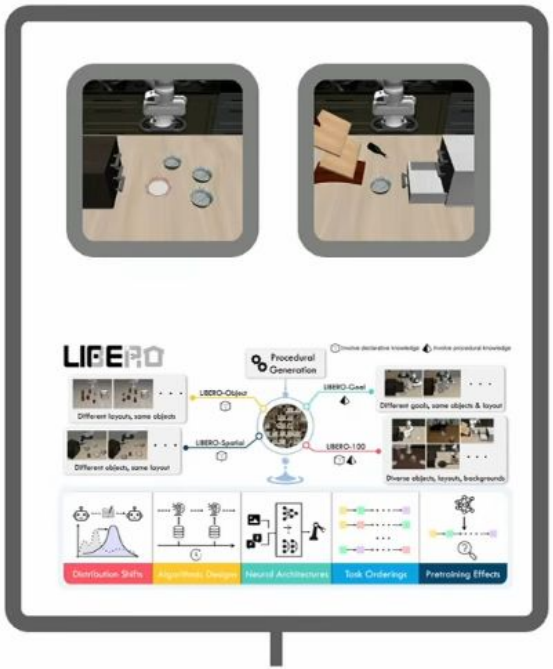
1. Better representation learning

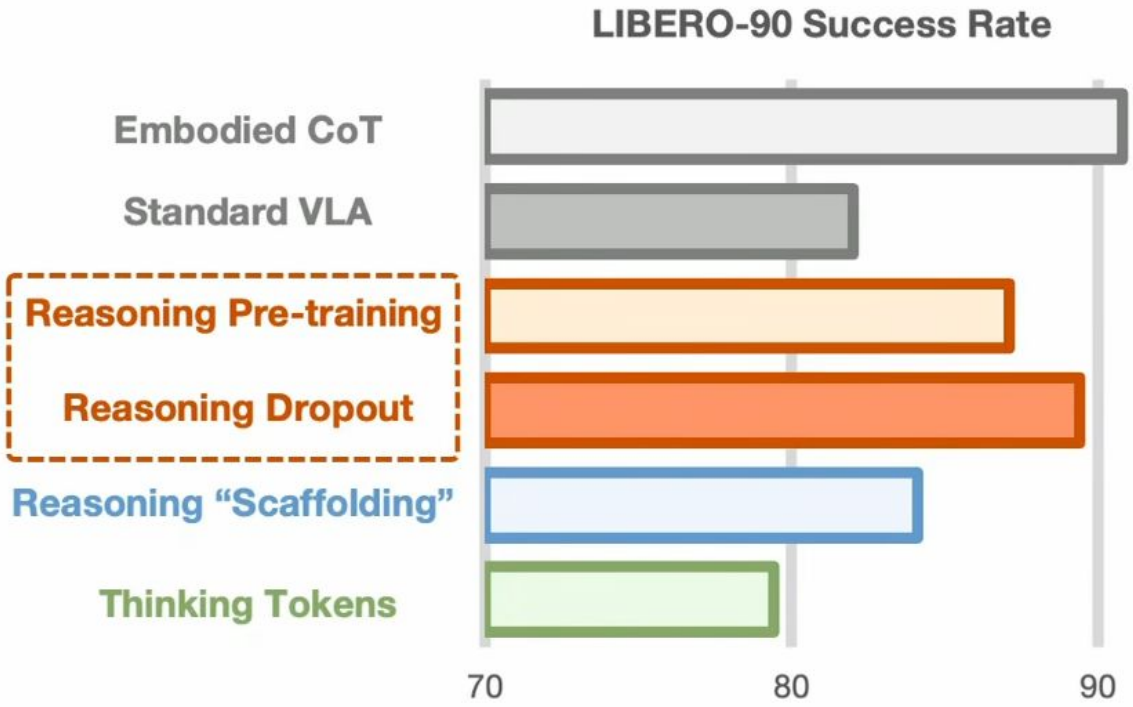
2. Improved learning "curriculum"

3. Increased policy expressivity

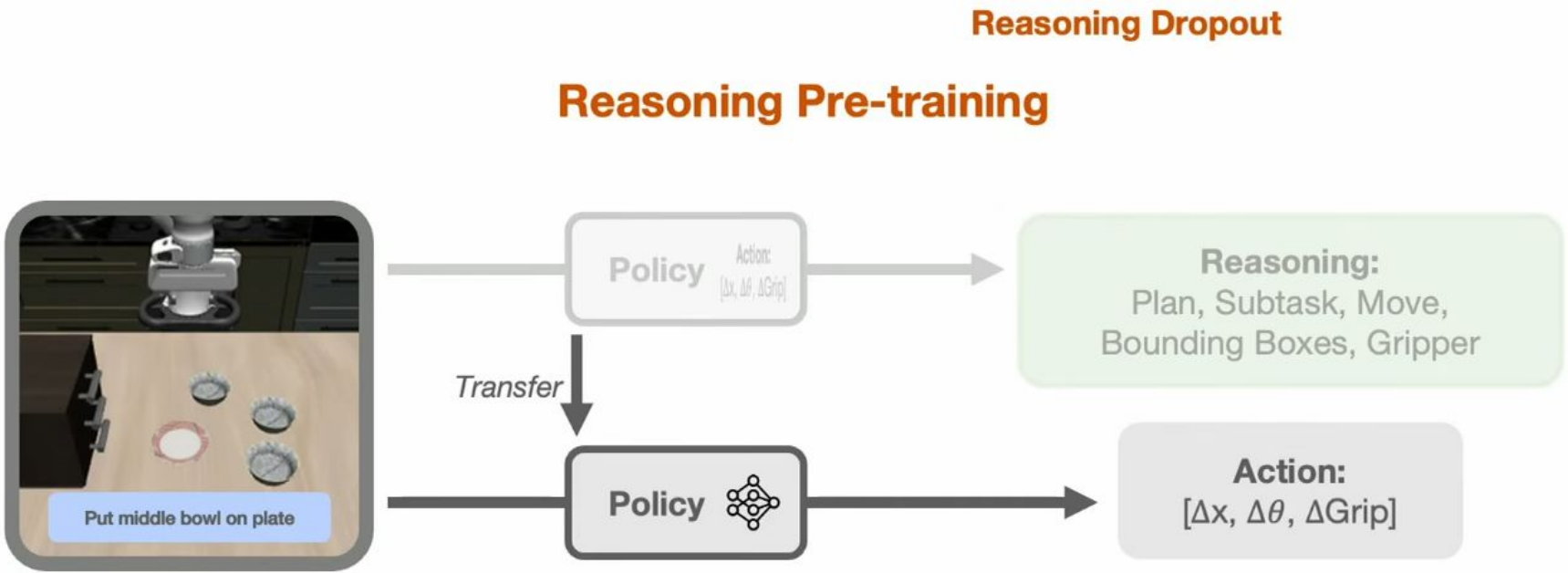


Results in LIBERO-90

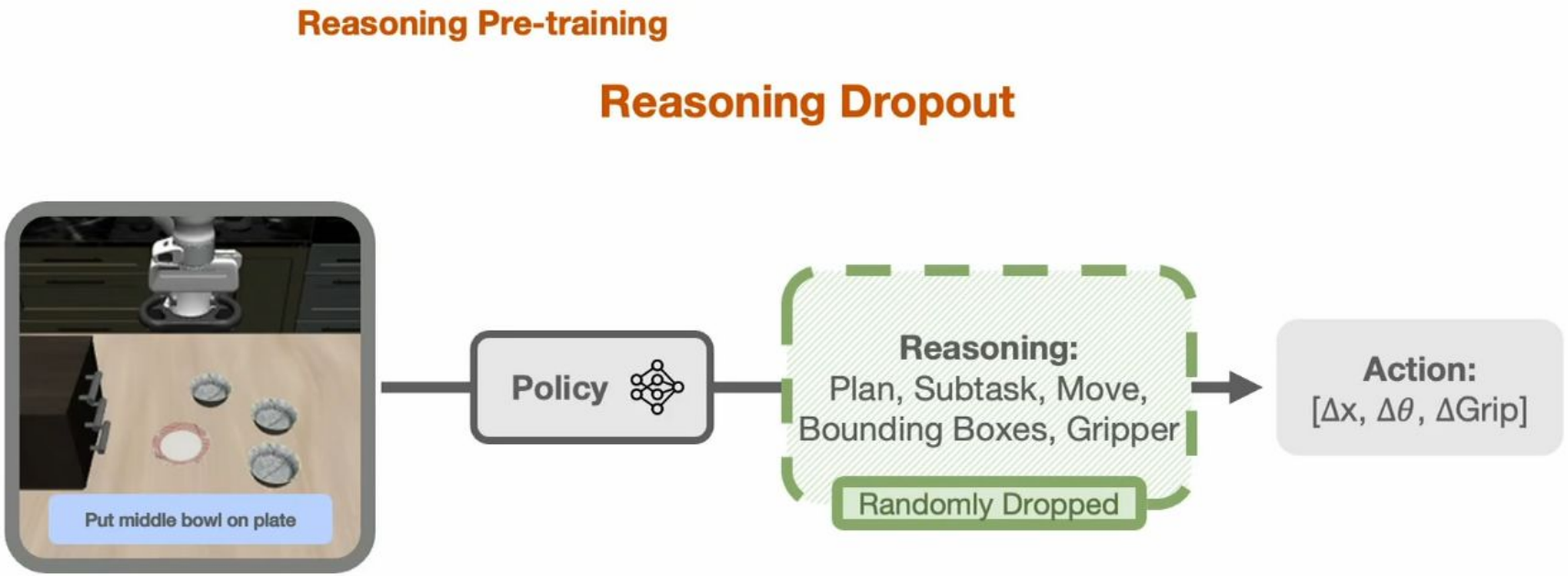




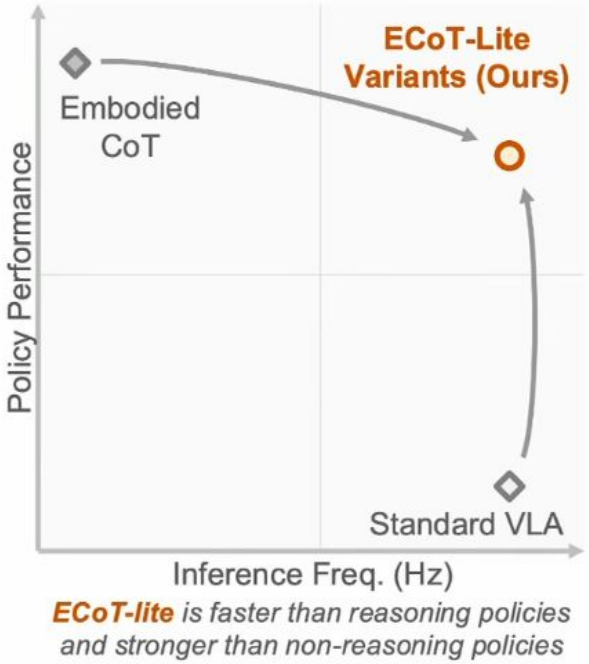
Embodied Chain-of-Thought Lite



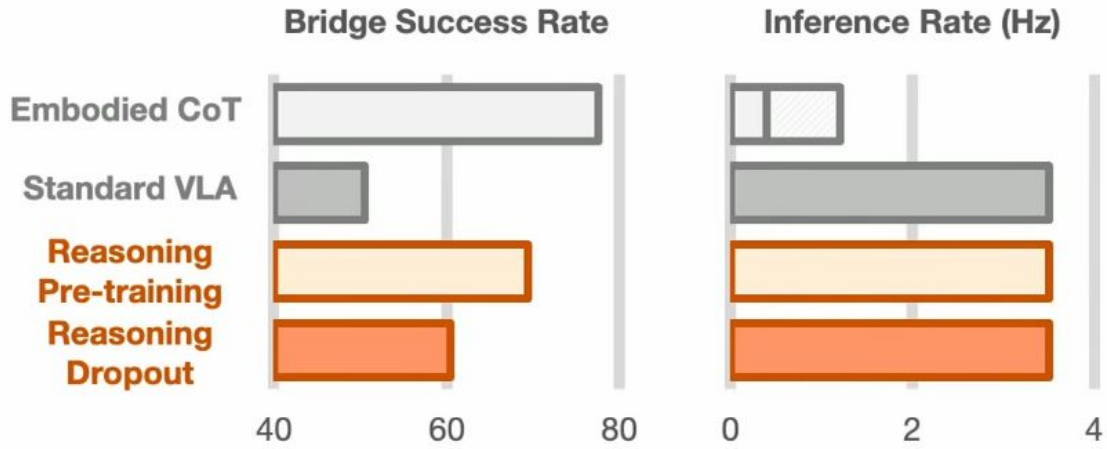
Embodied Chain-of-Thought Lite



Embodied Chain-of-Thought Lite: Results

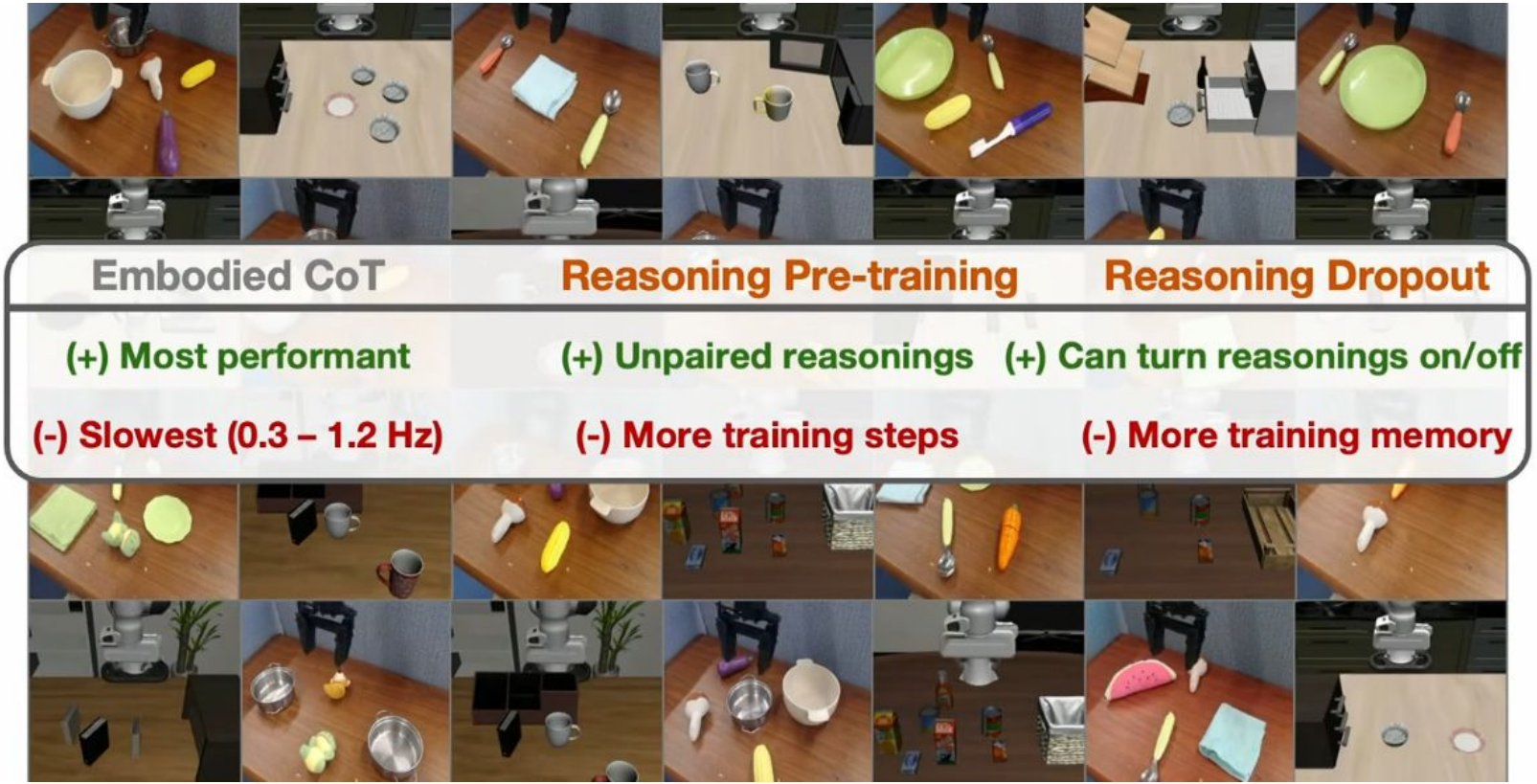


ECoT-lite *is faster than reasoning policies and stronger than non-reasoning policies*

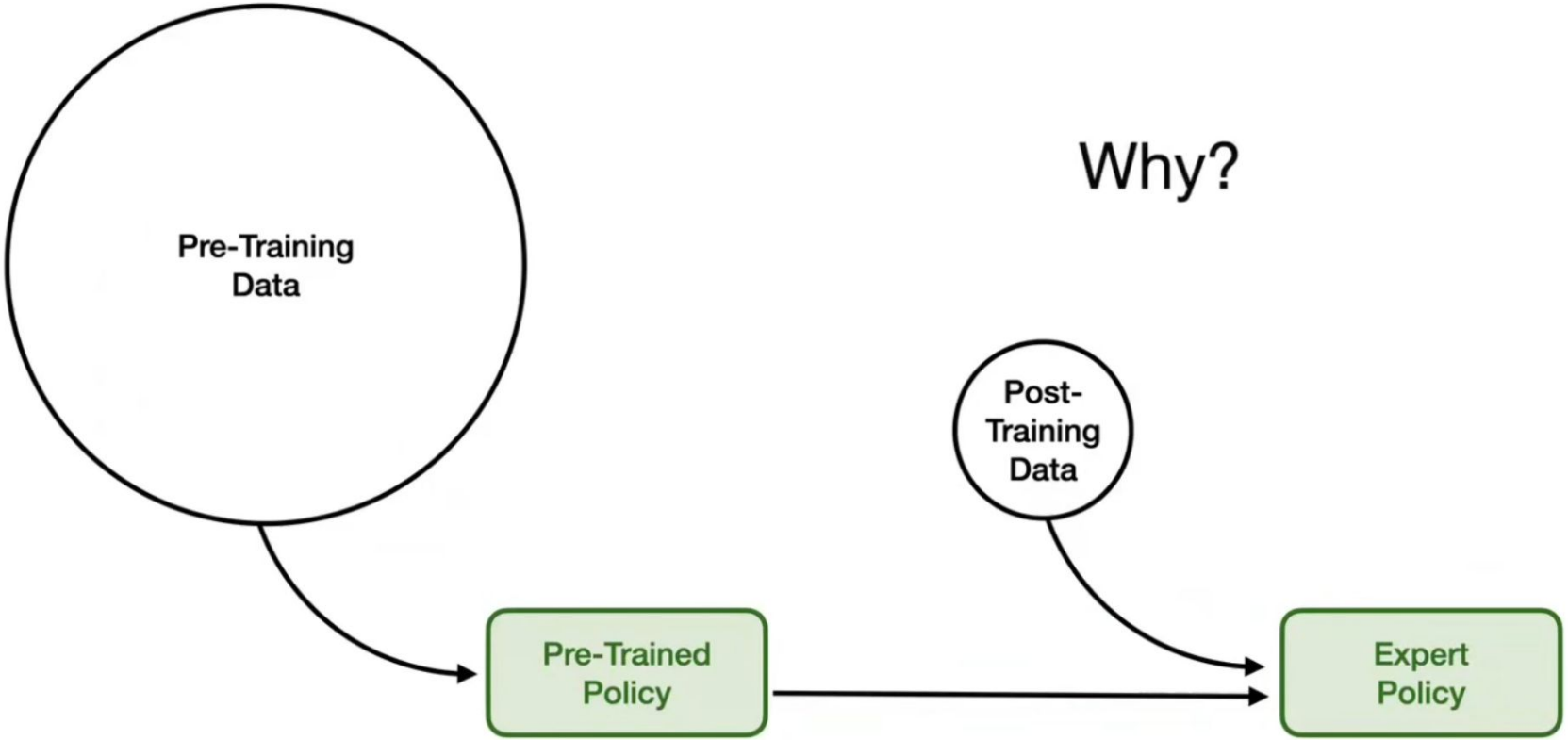


More Performant than Standard VLAs *without* sacrificing speed!

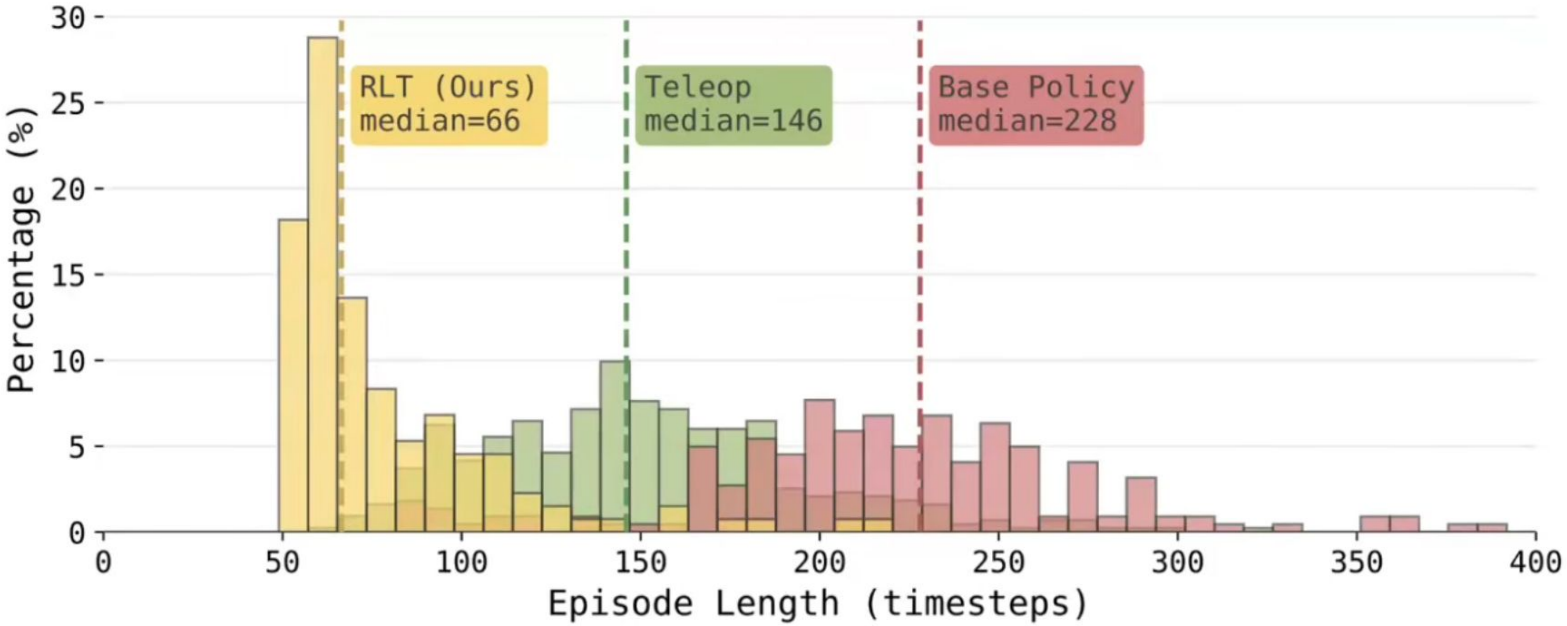
Embodied Chain-of-Thought Lite: Results



Why do we need Post-Training?



Data can be very diverse and multimodal



Fitting diverse data naively is hard + gives average performance in the best case

Solving hard long-horizon tasks



Generalization + Performance

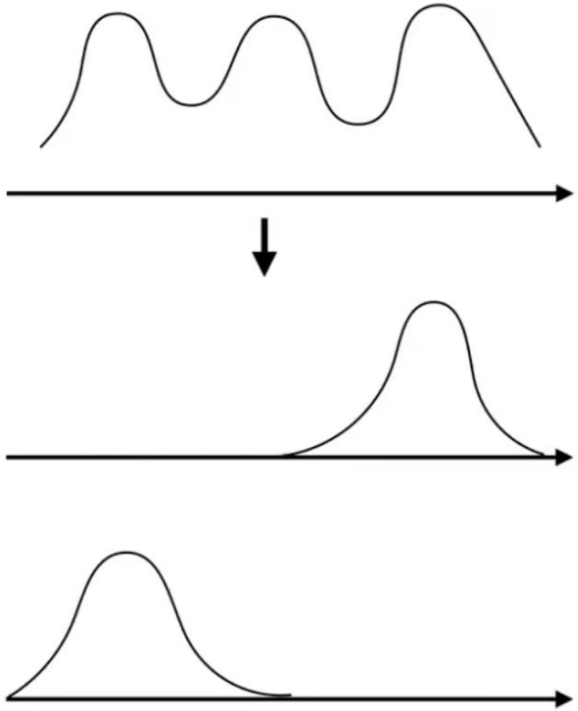
Pre-Training Post-Training

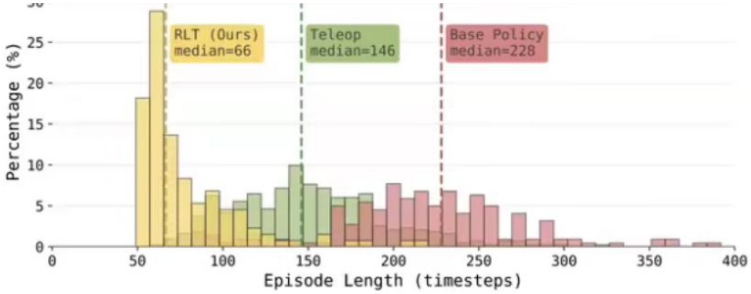
Fitting diverse data

$p(a \mid o)$

$p(a \mid o, c_1)$

$p(a \mid o, c_2)$



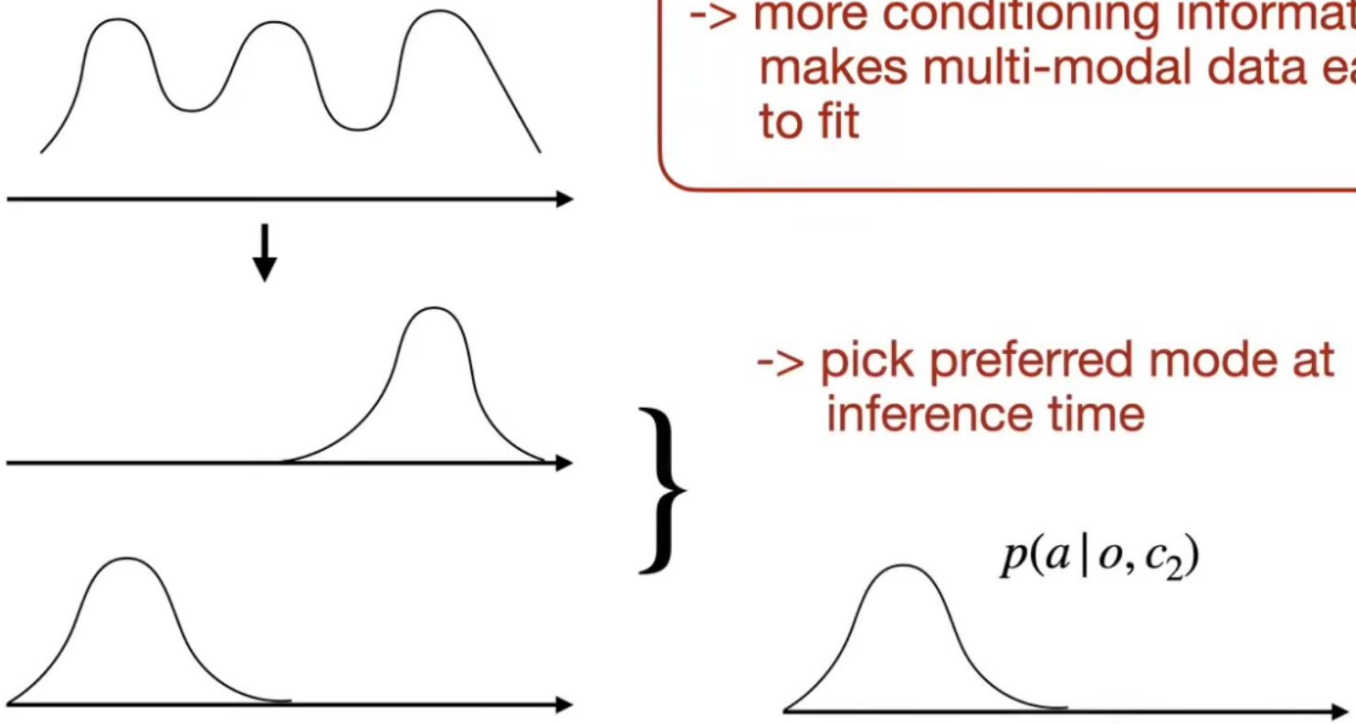


Fitting diverse data

$p(a \mid o)$

$p(a \mid o, c_1)$

$p(a \mid o, c_2)$



-> more conditioning information makes multi-modal data easier to fit

-> pick preferred mode at inference time

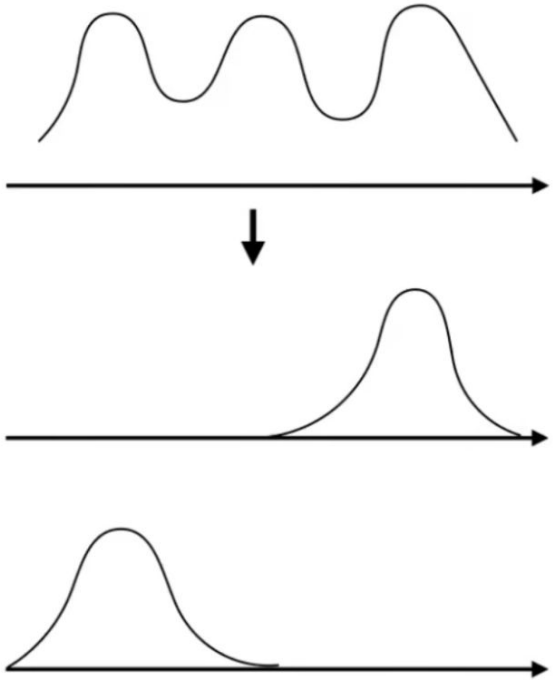
$p(a \mid o, c_2)$

Fitting diverse data

$p(a \mid o)$

$p(a \mid o, c_1)$

$p(a \mid o, c_2)$



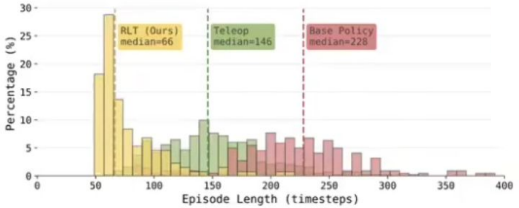


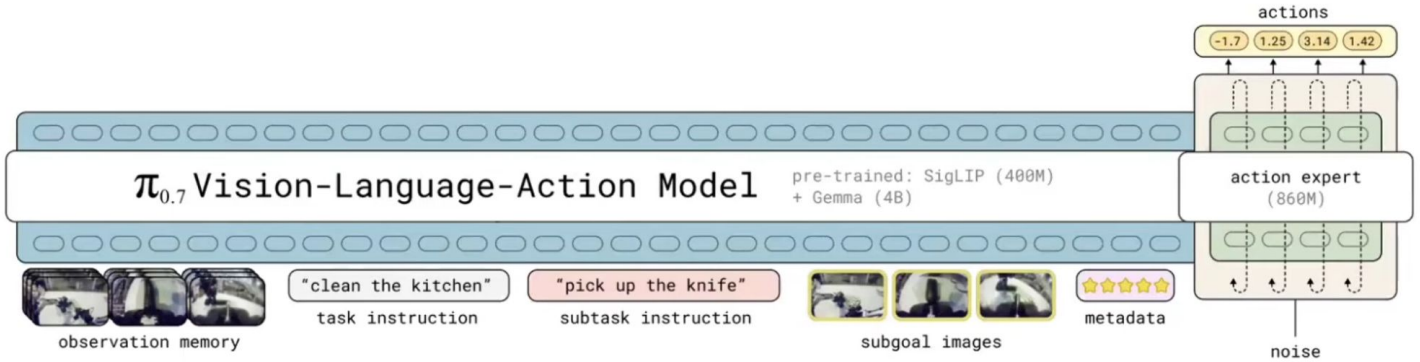
"A blue dog"



"A blue and white dog sitting on a green lawn, facing the camera, looking happy with its mouth slightly opened. A blurred sunset in the distance."

Idea: Diverse Conditioning

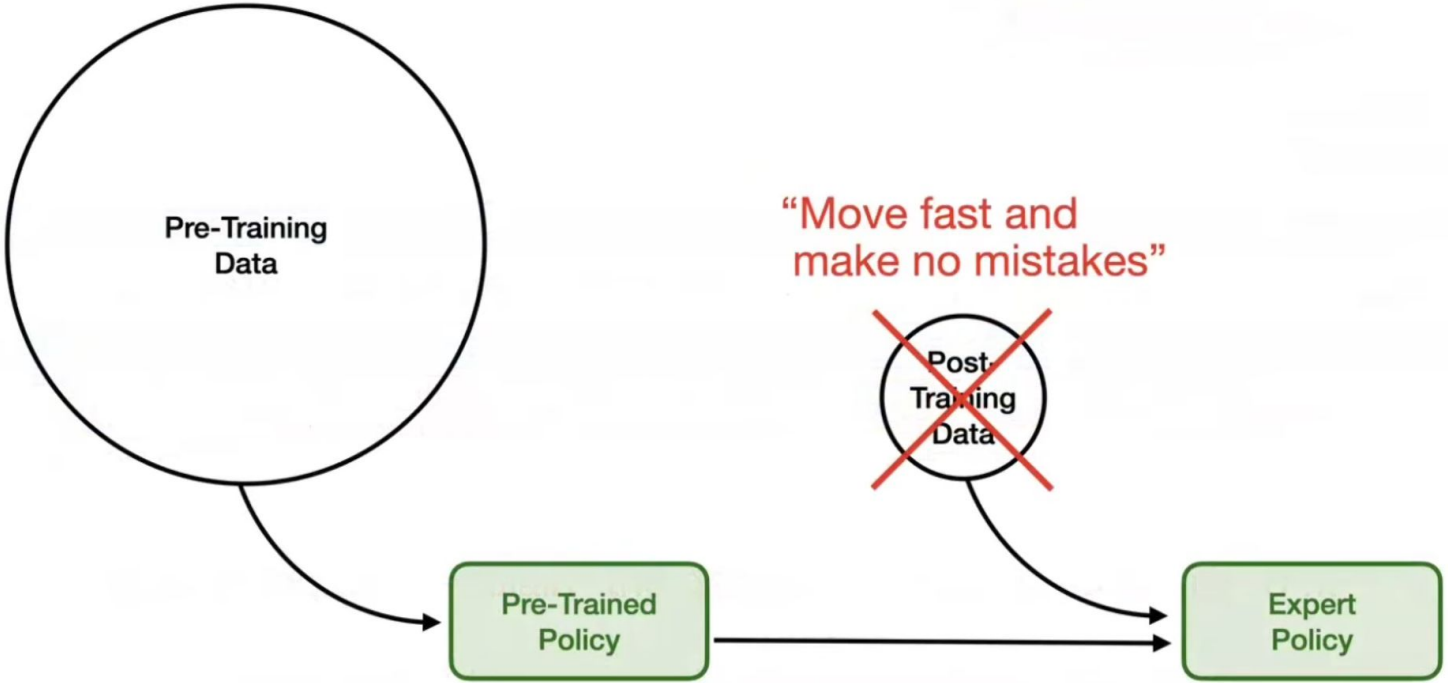




Conditioning can “explain” the data to the model —> easier to fit

We can “post-train” the policy at inference time by pulling out the desired behavior via targeted conditioning

Single-Stage Policy Training Pipeline



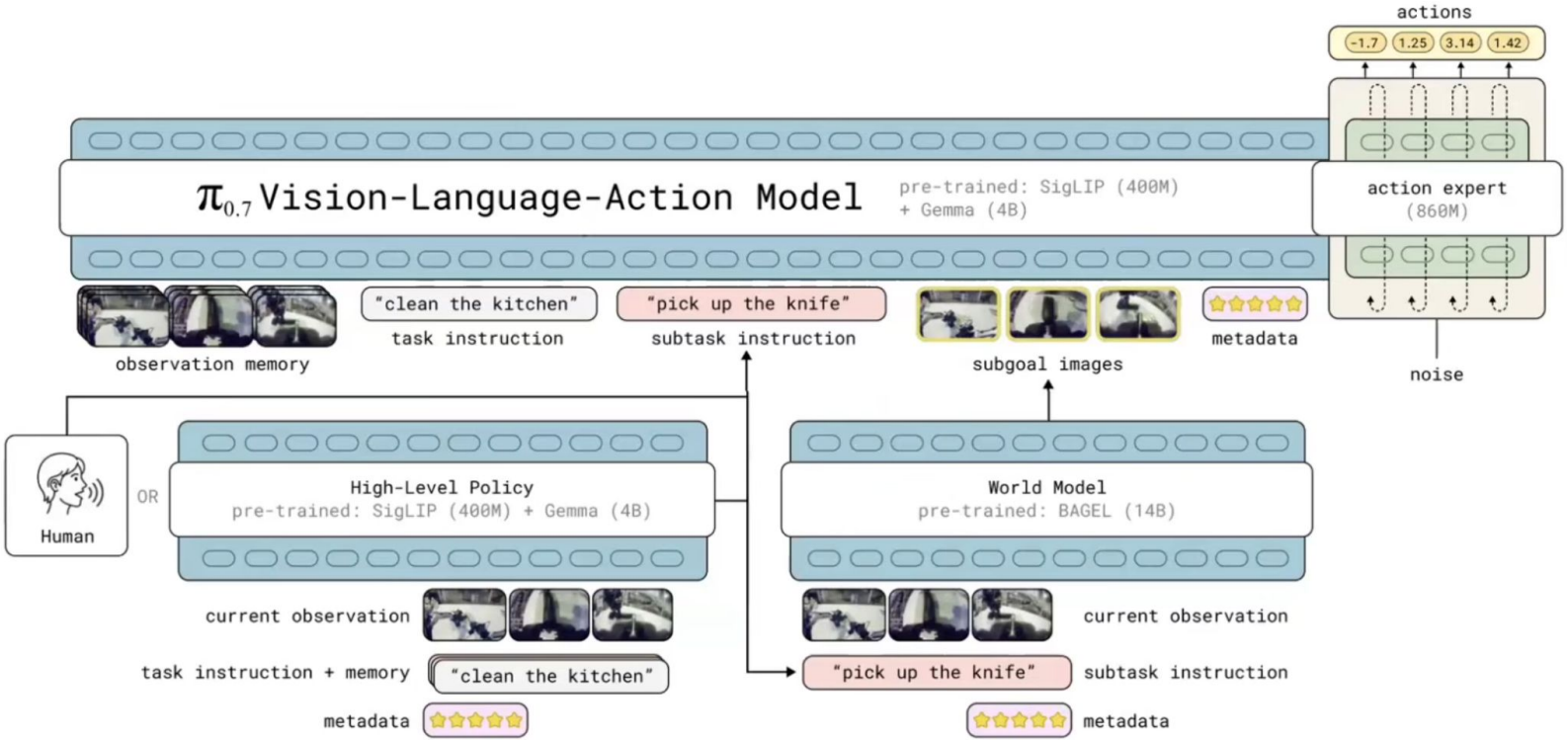
Coaching

“Load the sweet potato in the air fryer”

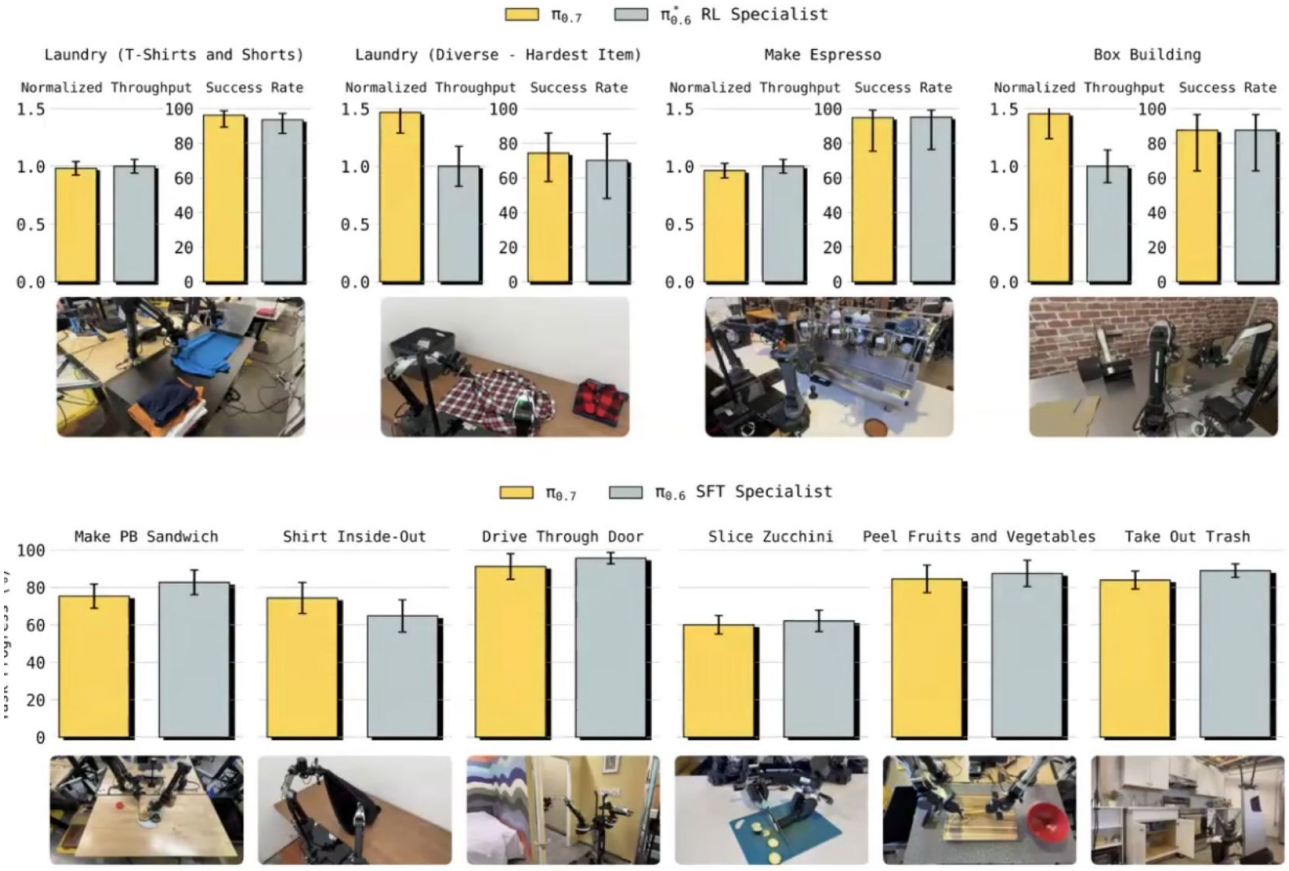




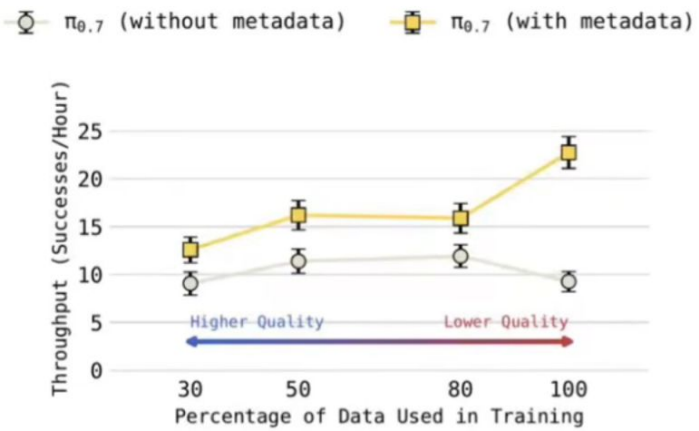
Idea: Diverse Conditioning



Idea: Diverse Conditioning



Learn from Mixed-Quality Experience





Steerable Vision-Language-Action Policies for Embodied Reasoning and Hierarchical Control

William Chen[1], Jagdeep Singh Bhatia[1], Catherine Glossop[1], Nikhil Mathihalli[1], Ria Doshi[2], Andy Tang[2], Danny Driess[3], Karl Pertsch[3], Sergey Levine[1]
[1]U.C. Berkeley, [2]Stanford University, [3]Physical Intelligence

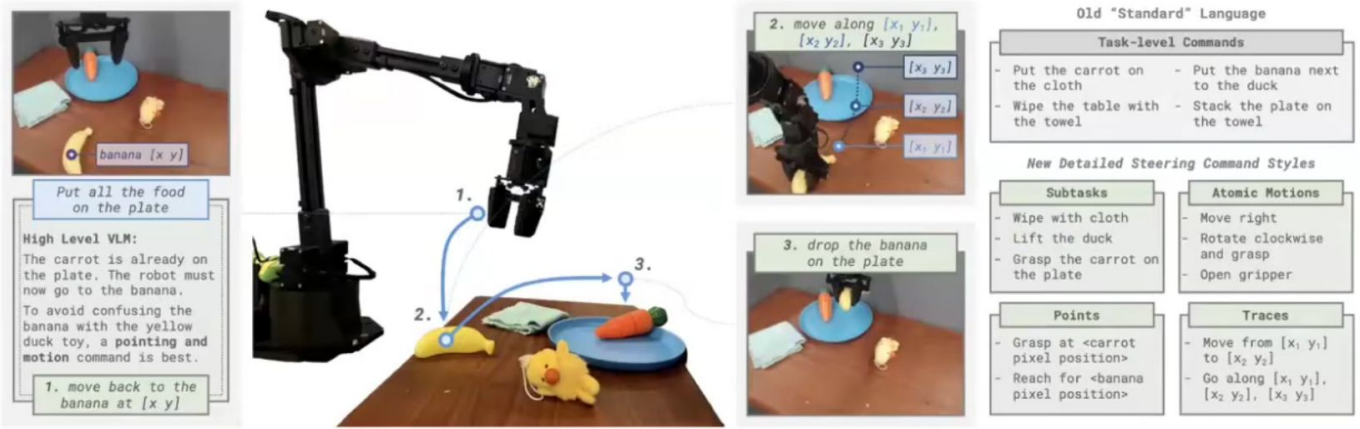
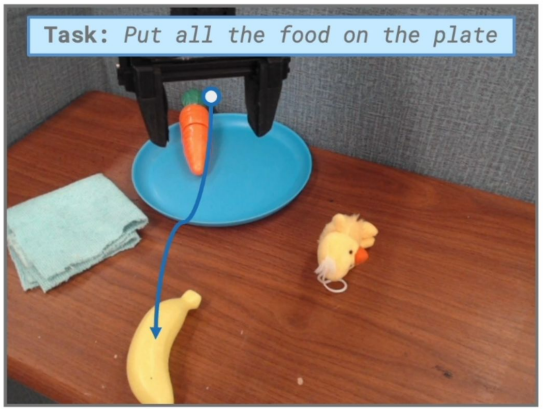


Fig. 1: We propose **Steerable Policies**: vision-language-action models that can robustly follow a wide range of detailed commands (green boxes on right), going beyond usual task language to include instructions such as motions or pixel coordinates of the gripper and objects. The flexibility afforded by Steerable Policies enables substantially improved transfer of VLMs' pretrained reasoning, semantic knowledge, and in-context learning skills to generalist robotic control in hierarchical systems.

Low-level steerability is a bottleneck



Observations

Vision-Language Model (VLM)

Commands

Put banana on plate

Vision-Language-Action Model (VLA)

Actions

Move to the duck toy

Vision-Language Model (VLM)

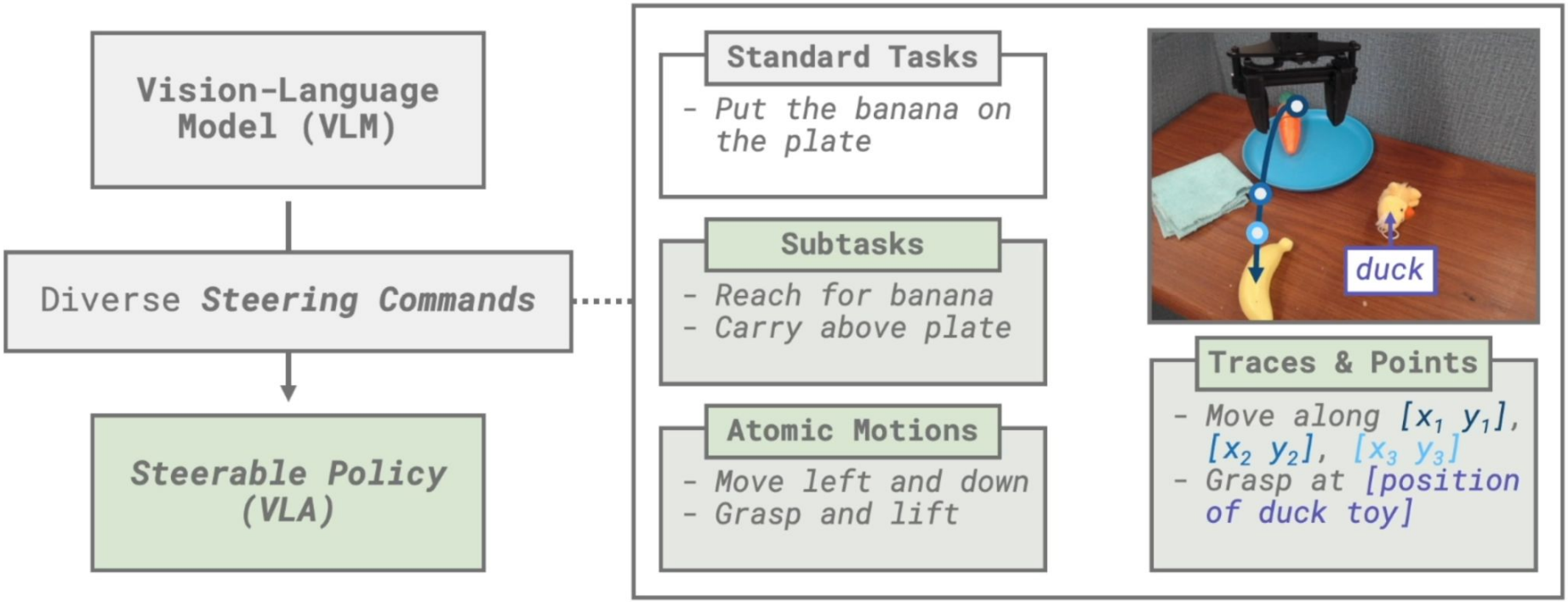
Commands

What's a duck?

Vision-Language-Action Model (VLA)

Must be able to execute *all* behaviors the VLM might produce!

Including Subtasks and Motions



Label robot behaviors

1. Extracting grounded features

2. Compiling into steering commands

3. Training with new commands

Put the banana on the plate



Bounding Boxes

Gripper Traces

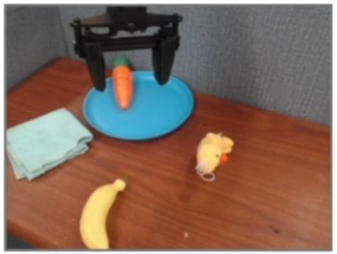
Subtasks

Reach for banana	*Carry banana above plate*	*Drop banana on plate*	*Carry banana above plate*

Label robot behaviors

1. Extracting grounded features | **2. Compiling into steering commands** | 3. Training with new commands

Put the banana on the plate



Bounding Boxes

Gripper Traces

Reach for banana

API-based VLM

Move toward the banana

Move back and down

Reach back along $[x_1\ y_1]$, $[x_2\ y_2]$, $[x_3\ y_3]$

Go to [banana's x y]

Label robot behaviors

1. Extracting grounded features
2. Compiling into steering commands
3. Training with new commands



Put the banana on the plate

Standard Behavioral Cloning

VLA

Reach back along $[x_1\ y_1]$, $[x_2\ y_2]$, $[x_3\ y_3]$



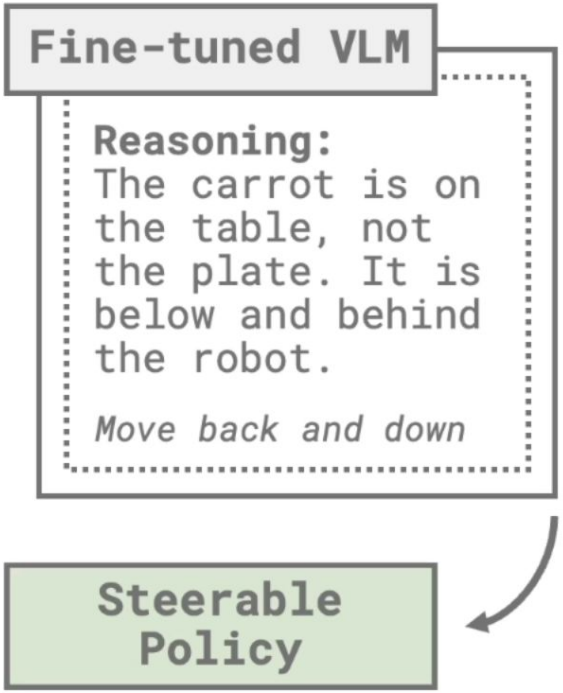
Actions

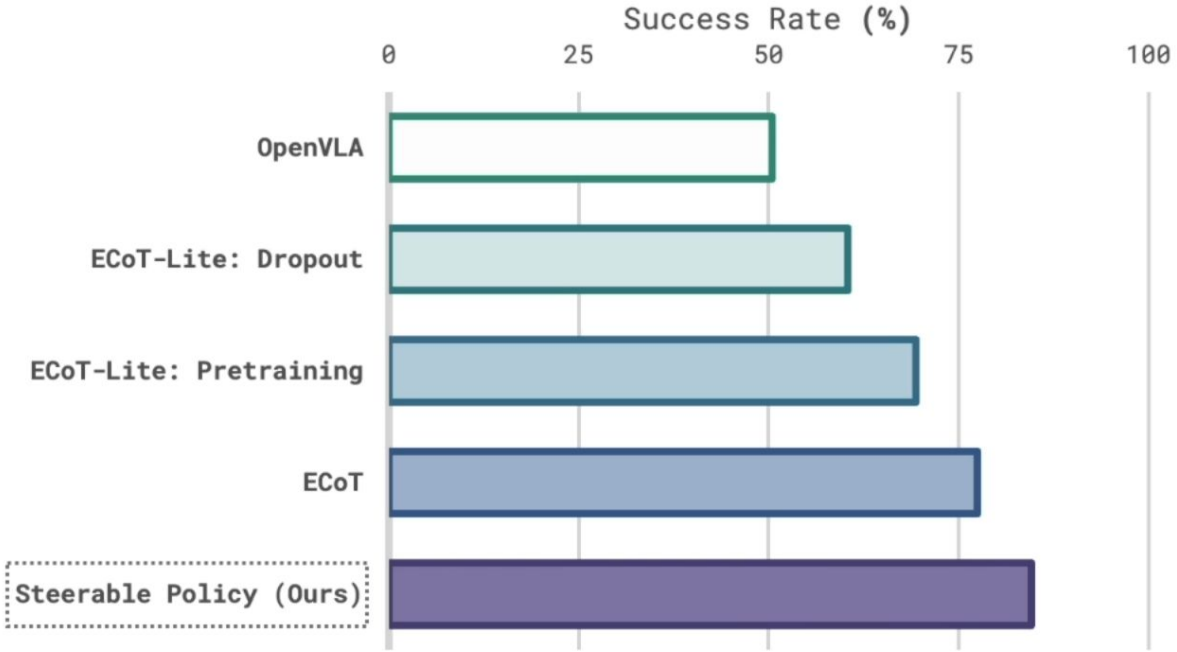
Move toward the banana

Move back and down

Go to [banana's x y]

Results





Our method outperforms past reasoning VLA methods!

Results

Embodied Reasoning

In-context Learning with Steerable Policies



Subtask *steering command* fails

Prior subtask commands names resulted in targeting the wrong object. → **A pointing command would be good instead.**

VLM adjusts steering command



Results in success!

Regular (non-steerable) policies cannot adjust command in this way!

Results

Embodied Reasoning

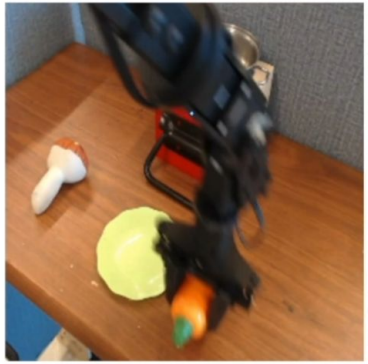


Step: 22

Rationale:
The robot needs to place the watermelon on the towel. The robot's gripper is currently empty, and the watermelon is the object with a pink and blue pattern on the table. The gripper is to the right of the watermelon, so the robot needs to move left to reach the watermelon. To properly grasp the watermelon, the gripper may also need to rotate counterclockwise to align its fingers and the object. Once aligned, the robot can stop moving and prepare to grasp the watermelon.

Steering Command:
move to [151, 72], [139, 58], [128, 48], [112, 48].

Put the watermelon on the towel



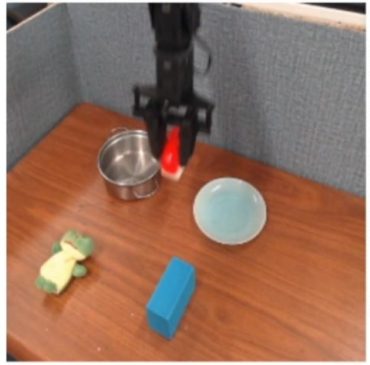
Step: 26

Rationale:
The robot's gripper is currently holding a carrot. To put the carrot in the pot, the robot first needs to lift the carrot slightly to avoid dragging it across the surface. The pot is located in front of the gripper, slightly to the left. To lift the carrot effectively, the robot should move forward and slightly to the left, ensuring the carrot clears the other objects on the table.

Steering Command:
move the carrot up away from the pot at [133, 79].

Put the carrot in the pot

In-context Learning

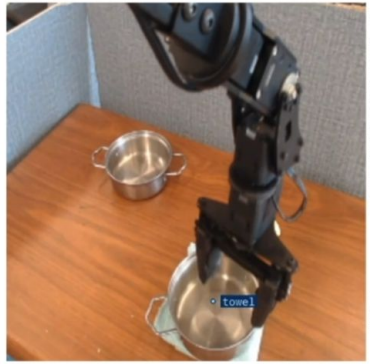


Step: 44

In-context Reasoning:
Step 1: In the current observation, the robot has successfully picked up the red sushi from the plate and is now holding it directly above the metal pot. The plate is currently empty, and the blue block remains on the table.
Step 2: The robot needs to release the red sushi into the pot to clear its gripper for the next task.
Step 3: Since the robot is already positioned correctly above the pot with the sushi, a high-level command is the most efficient way to complete the release action. This level of abstraction is reliable for simple terminal actions like dropping an object.

Steering Command:
drop the red sushi in the pot

Make the blue block the only thing on the plate



Step: 31

In-context Reasoning:
Step 1: The robot has successfully grasped and lifted the pot that was at the top of the scene. The second pot, banana, and light blue towel remain on the table.
Step 2: The robot needs to move the held pot and place it onto the towel to begin the stacking process.
Step 3: A combination instruction is appropriate here to specify both the action and the target location clearly. Using the towel as a reference point with position-based notation will help ensure the robot moves to the correct area.

Steering Command:
move the pot in the gripper to the towel at [147, 213]

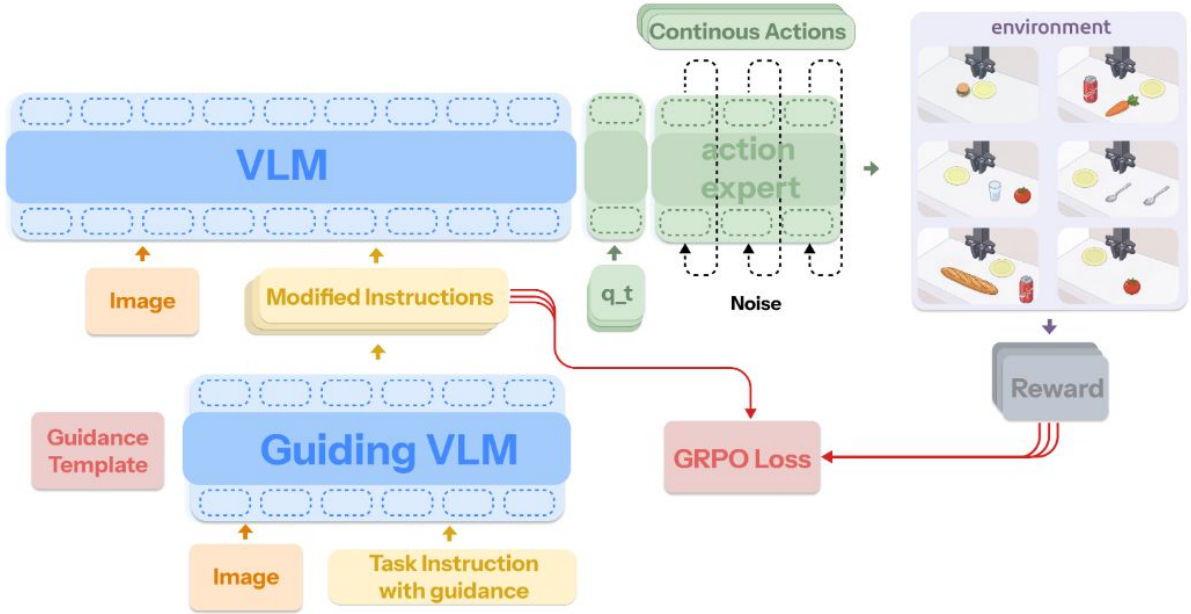
Stack the pots on the towel

VLA Grounder: Language-Conditioning Space Optimization for Black-Box VLA Models



Trajectories comparison on RL4VLA MultiObject with frozen π0. The default command underspecifies the object and causes unstable motion, while the rewritten command adds visible grounding cues and yields a more direct successful trajectory.

VLA Grounder: Language-Conditioning Space Optimization for Black-Box VLA Models



Architecture and training. A language-conditioning space policy maps the task instruction, scene image, and a failure-derived command-space prior to a VLA-grounded command, which is executed by a frozen VLA. Only this policy is trained with RL from sparse task-completion rewards; VLA weights remain frozen.

VLA Grounder: Language-Conditioning Space Optimization for Black-Box VLA Models

Full Prompt and Example Output

Exact prompt prior, runtime scene-conditioned input, and concrete model response. The language policy may reason in <think>, but only <answer> is sent to the frozen VLA policy.

fixed prior | image + original task | <think> may reason | <answer> goes to VLA

role: system

fixed across tasks

You are a language translation assistant for robot manipulation tasks. You translate human instructions into commands a Vision-Language-Action (VLA) robot can act on.

You will be given:
- A user-provided instruction describing a manipulation goal and an image of the scene.

Your task is to:
1. Understand the meaning of the original instruction.
2. Correctly identify the object to pick and the target location to place it, then reason about how the robot visually perceives each — these may not be the same as your own understanding.
3. Generate a rephrased command that helps the robot locate and act on the target.

Keep in mind: the VLA perceives the scene visually and may not recognize objects the same way you do. Your translation must bridge that perceptual gap.

Feel free to explore.

role: user

current image and original instruction

[image: current scene]

Task: {txt}.
Generate a command for the robot to successfully complete the task.
When describing the object or the target, always reason from what the robot likely sees in the image, not from your own understanding of the object. The rules below are all applications of this.

Rules for the command:
1. Always keep the object name (e.g. 'carrot'), unless the object is unrecognizable to the robot — in that case replace the name with a visual description per Rule 7.
2. One short sentence. No line breaks, no numbering.
3. Try different grounding cues to help locate the object or the target — for example: spatial words (left, right, center, far, near), visual properties (color, size, shape), or relative position (closest, next to the cup). Try combining cues (e.g. 'far right', 'small red one on the left'). Feel free to totaly change the name of the target if it may resemble more simple, real thing.
4. Use only simple, everyday words.
5. If the object to pick or the placement target is something the robot is unlikely to recognize by name (abstract concept, brand, celebrity, symbol), describe it purely by its visual appearance in the scene — what the robot would actually see it as.

Think inside <think></think> tags following these steps:
1. Identify the object to pick and the placement target in the scene.
2. Reason about how the robot likely perceives each.
3. Decide how to rephrase the command.
Keep thinking process short, no more than 10 sentences.
Then give your final command inside <answer> your_command_here </answer> tags.
NOTE: only the command inside <answer></answer> will be passed to VLA!

model output example

reasoning plus final command

MODEL OUTPUT: REASONING
<think>
The user wants me to translate the instruction "put carrot on sunrise icon" into a command for a robot.
1. Identify the object to pick: There is an orange object with a green top on the table. It looks like a carrot. The robot might see it as an orange carrot-shaped object or just carrot. Given Rule 1, I should keep carrot if recognizable, which it likely is.
2. Identify the placement target: There are three white cards on the table. Left card: grey squiggly lines. Center card: yellow semi-circle with rays behind horizontal lines; this looks like a sun setting or rising. Right card: cloud with blue rain drops. The target is the center card with the sun symbol. The robot sees a white square card with a yellow sun symbol.
3. Reasoning about perception: The robot sees a carrot lying on the table. The robot sees three cards. The one in the middle has a yellow sun drawing. The command needs to direct the robot to move the carrot to that specific card.
4. Drafting the command: "Put the carrot on the sun card" is a bit vague. "Pick up the orange carrot and place it on the white square with the yellow sun" is more descriptive. "Place the carrot on the center card with the sun picture" adds a spatial cue. "Put the carrot on the card with the yellow sun and black rays" is very specific.
The rule asks for grounding cues: spatial words and visual properties. "Sunrise icon" might be abstract for a robot, while "white card with the yellow sun" is concrete. The object name carrot should remain because it is recognizable. Final choice should be one short sentence with simple words.
</think>

<answer>
Place the carrot on the white card with the yellow sun symbol.
</answer>

VLA Grounder: Language-Conditioning Space Optimization for Black-Box VLA Models

	π_0				OpenVLA			
Task	orig	Guidance w/o GRPO	GRPO w/o Guidance	GRPO + Guidance	orig	Guidance w/o GRPO	GRPO w/o Guidance	GRPO + Guidance
VL-Think								
Arrow	6.2 ± 1.3	7.3 ± 2.7	$\mathbf{37.5 \pm 6.4}$	36.5 ± 3.2	13.5 ± 3.2	21.4 ± 2.0	34.9 ± 6.0	$\mathbf{62.5 \pm 1.3}$
Color	45.3 ± 7.1	35.9 ± 2.2	$\mathbf{49.5 \pm 4.1}$	44.8 ± 7.0	55.7 ± 8.7	63.5 ± 4.5	75.5 ± 3.2	$\mathbf{82.8 \pm 2.6}$
Laundry	16.1 ± 0.7	12.5 ± 1.3	$\mathbf{31.2 \pm 2.2}$	26.6 ± 2.6	15.6 ± 2.6	21.9 ± 5.6	28.7 ± 4.8	$\mathbf{37.5 \pm 3.8}$
Public Info	14.6 ± 1.5	20.3 ± 2.2	34.4 ± 2.2	$\mathbf{43.2 \pm 2.9}$	21.4 ± 3.7	27.1 ± 3.2	40.6 ± 3.4	$\mathbf{65.6 \pm 3.4}$
Shape	20.3 ± 5.8	34.9 ± 4.5	37.0 ± 6.4	$\mathbf{49.5 \pm 6.0}$	27.6 ± 1.5	49.0 ± 7.7	68.2 ± 3.2	$\mathbf{74.5 \pm 6.4}$
Traffic	6.2 ± 1.3	14.1 ± 4.6	35.9 ± 8.9	$\mathbf{37.0 \pm 10.2}$	16.7 ± 2.7	28.1 ± 4.6	42.2 ± 7.8	$\mathbf{63.0 \pm 4.8}$
Weather	19.3 ± 2.7	20.8 ± 3.7	35.4 ± 5.9	$\mathbf{42.2 \pm 3.4}$	20.3 ± 4.6	31.3 ± 3.4	41.2 ± 5.3	$\mathbf{61.5 \pm 3.9}$
Avg.	18.3	20.8	37.3	**40.0**	24.4	34.6	47.3	**63.9**
RL4VLA								
MultiPlate	16.7 ± 7.0	14.6 ± 6.0	19.3 ± 6.0	$\mathbf{20.3 \pm 2.6}$	54.7 ± 2.2	56.3 ± 4.6	62.5 ± 2.1	$\mathbf{64.6 \pm 5.2}$
MultiCarrot	24.5 ± 0.7	30.2 ± 0.7	29.2 ± 2.7	$\mathbf{33.3 \pm 4.8}$	59.4 ± 5.9	60.9 ± 2.6	61.5 ± 4.8	$\mathbf{63.5 \pm 5.3}$
Avg.	20.6	22.4	24.2	**26.8**	57.0	58.6	62.0	**64.0**

Ablation on frozen π0 over VL-Think benchmark. “No reasoning” denotes a generic command-rewriting policy; “reasoning” denotes the structured guidance used by our method.

Takeaways

1. “Explaining” your data well enables on-the-fly “post-training” via **conditioning**

2. With appropriate conditioning, policies can match **expert performance and generalize** broader

3. This is the foundation for pushing **long-horizon** autonomy