Probing a Vision-Language-Action Model for Symbolic States and Integration into a Cognitive Architecture

Hong Lu*
Department of Computer Science
Tufts University
Medford, MA, USA
Hlu07@tufts.edu

Hengxu Li*
Department of Computer Science
Tufts University
Medford, MA, USA
Hengxu.Li@tufts.edu

Prithviraj Singh Shahani*
Department of Computer Science
Tufts University
Medford, MA, USA
Prithviraj_Singh.Shahani@tufts.edu

Stephanie Herbers
Department of Computer Science
Tufts University
Medford, MA, USA
Stephanie.Herbers@tufts.edu

Matthias Scheutz
Department of Computer Science
Tufts University
Medford, MA, USA
Matthias.Scheutz@tufts.edu

***Abstract*—Vision-language-action (VLA) models hold promise as generalist robotics solutions by translating visual and linguistic inputs into robot actions, yet they lack reliability due to their black-box nature and sensitivity to environmental changes. In contrast, cognitive architectures (CA) excel in symbolic reasoning and state monitoring but are constrained by rigid predefined execution. This work bridges these approaches by probing OpenVLA's hidden layers to uncover symbolic representations of object properties, relations, and action states, enabling integration with a CA for enhanced interpretability and robustness. Through experiments on LIBERO-spatial pick-and-place tasks, we analyze the encoding of symbolic states across different layers of OpenVLA's Llama backbone. Our probing results show consistently high accuracies (> 0.90) for both object and action states across most layers, though contrary to our hypotheses, we did not observe the expected pattern of object states being encoded earlier than action states. We demonstrate an integrated DIARC-OpenVLA system that leverages these symbolic representations for real-time state monitoring in the appendix, laying the foundation for more interpretable and reliable robotic manipulation.**

***Index Terms*—Vision Language Action Model, Symbolic States, Cognitive Architectures, Robotics**

*These authors contributed equally.

I. Introduction

A vision-language-action (VLA) model is a type of foundation model for robotics that takes in images and language commands as input and directly outputs robot actions [1], [2]. VLAs show promise in providing generalist robot policies across different scenarios and robotic platforms [1]. Recently, OpenVLA has emerged as a significant open-source VLA model, built on a Llama 2 language model backbone combined with a visual encoder that fuses pretrained features. Despite using only 7B parameters (7x fewer than comparable models), OpenVLA has demonstrated strong generalization capabilities across diverse manipulation tasks through its training on nearly one million real-world robot demonstrations [2].

However, recent evaluation of VLAs shows that they struggle with changes in environmental factors such as camera poses, lighting conditions, and the presence of unseen objects [3]. VLAs also lack reliability, particularly because of their opaque, black-box nature, which makes their internal workings challenging to interpret.

On the other hand, traditional Cognitive Architectures (CA), also known as symbolic architectures, excel in dependable, symbol-based reasoning but are constrained by their reliance on predefined rules and coded policy execution [4], [5]. Ideally, a CA could harness the versatility of generalist robotic policies and the multimodal capabilities offered by VLAs while maintaining vigilance over dynamic environmental changes during execution in safety-critical applications such as robotic manipulations.

In this work, we investigate whether and how OpenVLA encodes symbolic representations in its activation space through probing experiments. Our investigation aims to answer the following research questions:

- RQ1: To what extent can we decode object properties and relations (e.g., spatial relationships between objects) from OpenVLA's hidden layer activations?
- RQ2: Can we extract action-related concepts (e.g., grasp states, movement targets) from the model's activation patterns, and how do these compare to object-level representations?

To answer these questions, we train linear probes on different layers of OpenVLA to predict symbolic states during manipulation tasks. Based on prior work in language model probing [6] [7], we hypothesize that:

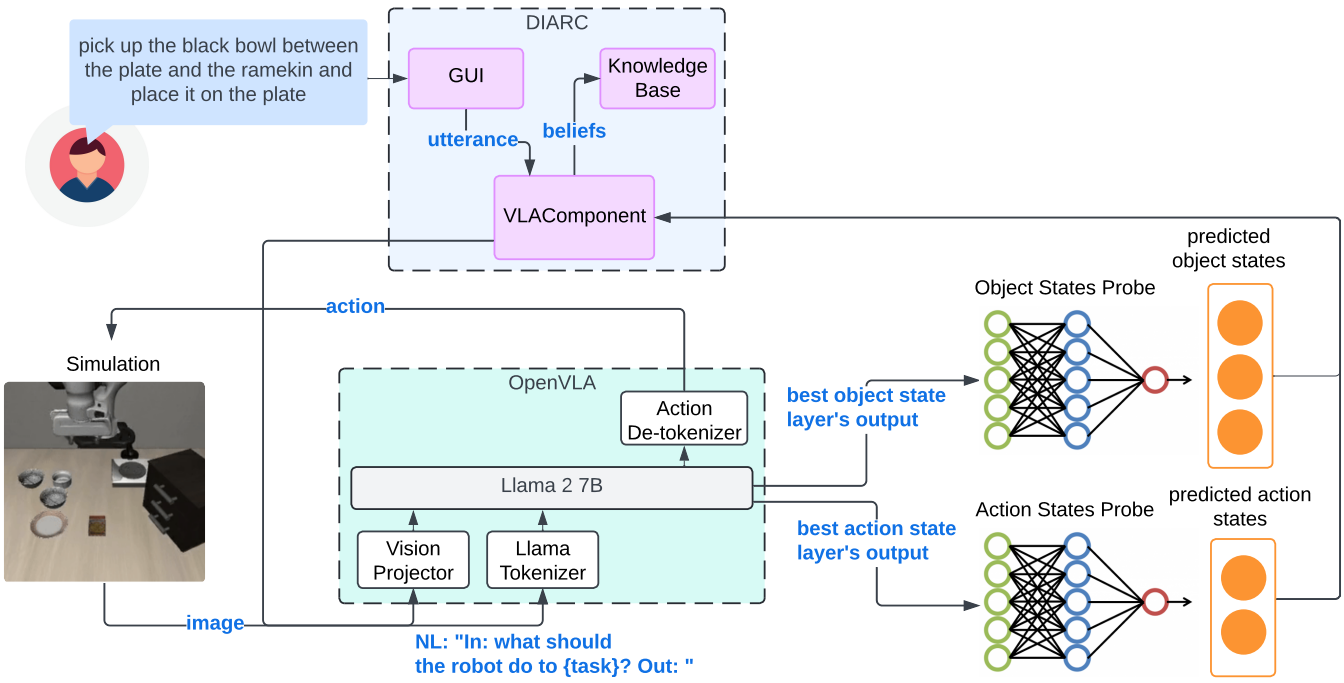

Fig. 1. **The DIARC - VLA - Probes System**. The user selects a natural language command in DIARC's Graphical User Interface (GUI). The VLAComponent in DIARC sends this command to OpenVLA. The probes receive two hidden layers' activations in OpenVLA's Llama backbone that encode the most object state and action state information respectively. The two best hidden layers are identified through the probing experiment described in Section IV. The probes predict the object state and the action state based on the hidden layers' activations at each timestep. The VLAComponent in DIARC updates DIARC's beliefs based on the predicted object state and action state.

- H1: Object states are primarily encoded in *earlier* layers, as these may encode basic visual and spatial properties.
- H2: Action-related concepts are encoded in *later* layers, where visual and language information has been integrated for action planning

To test these hypotheses, we conduct a probing experiment in which we examine *all 33 hidden layers* of OpenVLA for their capacity to predict object and action states. Figure 1 shows how we integrate the best-performing layers (for object vs. action states) into DIARC for real-time symbolic monitoring of the model's internal state.

II. Related Work

A. Cognitive Architecture - Foundational Model Integration

Existing works typically explore how CAs can be integrated with large language models (LLMs). For example, Wu et al. investigate whether the Llama-2 13B model encodes features that can predict expert decisions in a decision making task by training a linear classifier on the Llama model's last contextual embeddings to predict ACT-R's expert decision when the model is given ACT-R's strings of decision making traces as input; they further examine whether ACT-R's knowledge can be injected into the Llama model by fine-tuning a Llama-classifier system on ACT-R's expert decisions [8]. Bajaj et al. enhance ACT-R's analogical reasoning capabilities by building a natural language processing pipeline to automatically extract key entities, relationships, and attributes [9]. Once these key elements have been extracted from unstructured text, an LLM is prompted to convert the unstructured text into a structured format based on its key elements. ACT-R can utilize the structured knowledge for reasoning tasks downstream, significantly reducing the need for manual knowledge engineering. While Bajaj et al. propose to use LLMs to transform unstructured text into structured knowledge, Kirk et al. explore ways in which LLMs can be leveraged as a knowledge source for the task knowledge needed for successful task planning downstream [10]. They propose three approaches to knowledge extraction: indirect extraction in which the LLM's reponses are placed in a knowledge store that the cognitive agent accesses, direct extraction in which the agent directly queries the LLM and parses its output for structured knowledge, and direct knowledge encoding in which the LLM creates programs that are run as part of the cognitive agent's task pipeline.

To the best of our knowledge, no existing work has explored the integration of a CA with a VLA model. Realizing such an integration requires a method to "probe" the VLA's hidden representations for symbolic content. We next discuss relevant literature on foundational model probing, which informs our approach in extracting object and action states from OpenVLA.

B. Foundational Model Probing

Foundational models encode extensive knowledge derived from their internet-scale training data [11]. The increasing popularity of these models has drawn significant attention to the challenges of extracting and evaluating the knowledge they encode. An approach commonly used to evaluate LLMs is prompt-based probing in which an LLM is prompted to fill in the blanks on the prompt [12]. For example, Alivanistos et al. combine various prompting techniques to probe LLMs for possible objects of a triple where the subject and the relation are given [13], Wang et al. develop a method to automatically search sentences to construct optimal and readable prompts for probing certain knowledge [14], and Qi et al. propose a probing framework for multimodal LLMs that includes visual prompting, textual prompting, and extra knowledge prompting [15]. While prompt-based probing is intuitive and easy to execute, it lacks the layer-specific precision offered by linear probing. Furthermore, prompt-based probing is not applicable to vision-language-action models as they do not output language tokens. Linear probing on the other hand involves training a linear classifier on top of each frozen layer of a foundational model. Each classifier is tasked with predicting specific knowledge based on the output features of the corresponding frozen layer. For example, Li et al. train semantic probes to predict object properties and relations as they evolve throughout a discourse [16]. Similarly, Chen et al. use linear probes to evaluate the Llama model family's performance on higher-order tasks such as reasoning and calculation, comparing probe performance across layers and model sizes [6]. In our work, we extract symbolic representations of state changes similar to the approach described in Li et al [16] and we evaluate probing accuracies across hidden layers similar to the approach used by Chen et al [6].

III. Integrated Vision-Language-Action Model - Cognitive Architecture Overview

Figure 1 illustrates the high-level architecture of our DIARC–OpenVLA system. DIARC provides a cognitive architecture that manages symbolic reasoning and user interaction, while OpenVLA is a continuous policy that takes in images and language instructions to produce a 7D robot action. Internally, OpenVLA uses a Llama 2 7B backbone [7], which consists of 32 transformer blocks (often referred to as "layers") plus an initial embedding layer, yielding 33 distinct hidden states when indexed from 0 to 32 at runtime. Each hidden state is a 4096-dimensional vector. At a conceptual level, we combine these by: (1) routing user commands through DIARC to OpenVLA, (2) running OpenVLA in a LIBERO simulation environment to generate actions and extract hidden-layer embeddings, and (3) mapping those embeddings to symbolic states for DIARC's belief store. This pipeline leverages the expressiveness of a vision-language-action model while maintaining the reliability of a symbolic architecture. Subsection III-A details our real-time implementation, including the WebSocket interface and a React-based UI for visualization.

A. DIARC–OpenVLA-Probes Integration

The DIARC–OpenVLA-Probes integration bridges OpenVLA's continuous policy outputs and hidden-layer embeddings with DIARC's symbolic reasoning modules. Our approach requires minimal modification to OpenVLA itself: we intercept each inference call to extract the relevant hidden-layer activations and feed them to trained linear probes for symbolic state prediction, then pass those states back to DIARC in an automated fashion. Figure 1 provides a broad schematic, and Figure 6 shows the user interface in action.

a) VLAComponent and Symbolic Predicates.: At every timestep, OpenVLA predicts a 7D action $\Delta x, \Delta\theta, \Delta\text{Grip}$ given the current camera image and the user's natural-language instruction. In parallel, we run linear probes on the extracted hidden-layer embeddings to output arrays of 0/1 labels for object relations and action subgoals (e.g., *on(bowl, plate)* = 1, *grasped(bowl)* = 0). These arrays are sent to DIARC's VLAComponent, which converts them into DIARC's symbolic predicate format: *relation(object1, object2)*, *property(object)* for object states, and *action(object)* for action states. For example, a 1 in *grasped(bowl_1)* becomes *grasped(bowl_1)* in DIARC's knowledge store. DIARC can then leverage these discrete predicates to detect inconsistencies (e.g., a bowl cannot be both *on(bowl_1, plate_1)* and *inside(bowl_1, drawer_1)* at the same time), verify subgoals, or track overall task progress.

B. Simulated Pick-and-Place Tasks

LIBERO-object and LIBERO-spatial is each a suite of 10 pick-and-place tasks in the LIBERO simulation environment [17]. We choose this simulation environment for our OpenVLA evaluation as LIEBRO-obejct and LIBERO-spatial finetuned OpenVLA checkpoints are readily available for download. The LIBERO-object task suite consists of 10 pick-and-place tasks of the form "pick up the {target object} and place it in the basket". Figure 2 shows 4 frames of the OpenVLA performing the "pick up the cream cheese and place it in the basket" task. The LIBERO-spatial task suite consists of 10 pick-and-place tasks of the form "pick up the black bowl {spatial relations identifier} and place it on the plate" where the "spatial relations identifier" is filled with a natural language description of the target black bowl's spatial relations to its surrounding objects. For example, Figure 3 shows 4 frames of the OpenVLA performing the "pick up the black bowl between the plate and the ramekin and place it on the plate" task. Other LIBERO-spatial pick-and-place tasks include "pick up the black bowl next to the ramekin and place it on the plate" and "pick up the black bowl in the top drawer of the wooden cabinet and place it on the plate". Since the 10 pick-and-place tasks involve the same objects and the object initial placements remain the same across tasks except for the two black bowls, the natural language description of the target black bowl's spatial relations serves as an identifier.

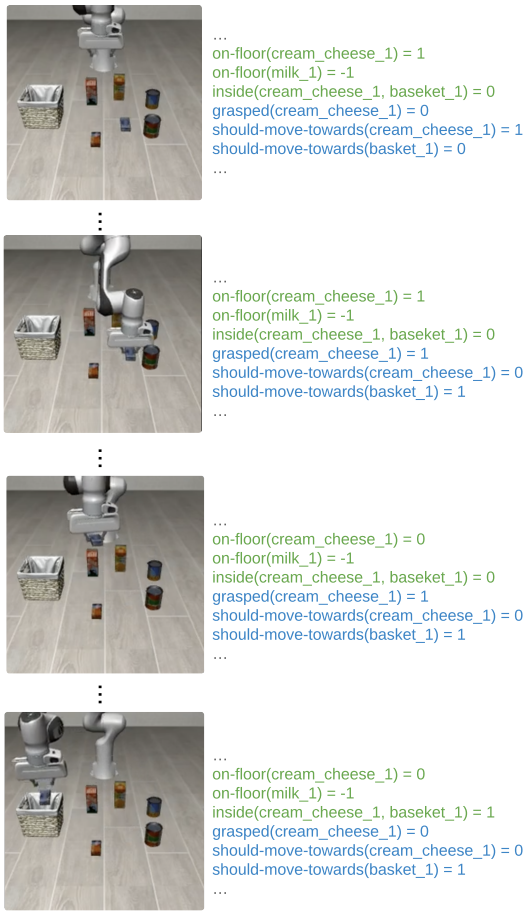

Fig. 2. **Example Labeled Object States and Action States in a LIBERO-object Pick-and-Place Trajectory.** Object states are shown in green and action states are shown in blue. The task is to "pick up the cream cheese and place it in the basket". Note that 0 represents false, 1 represents true, and -1 represents not applicable due to the object not being present. The first frame (top) is taken at the beginning of the episode where the cream cheese is directly on the floor and the robot needs to pick it up. The *on-floor(cream_cheese_1)* object relation and the *should-move-towards(cream_cheese_1)* action subgoal are detected to be true. Once the robot arm closes its grippers on the target black bowl in frame 2, the *grasped(cream_cheese_1)* action status is detected to be true. In frame 3, the *on-floor(cream_cheese_1)* object relation becomes false as the cream cheese is lifted off the floor. Finally, in frame 4, the *inside(cream_cheese_1, basket_1)* object relation becomes true, indicating that the task has been successfully completed.

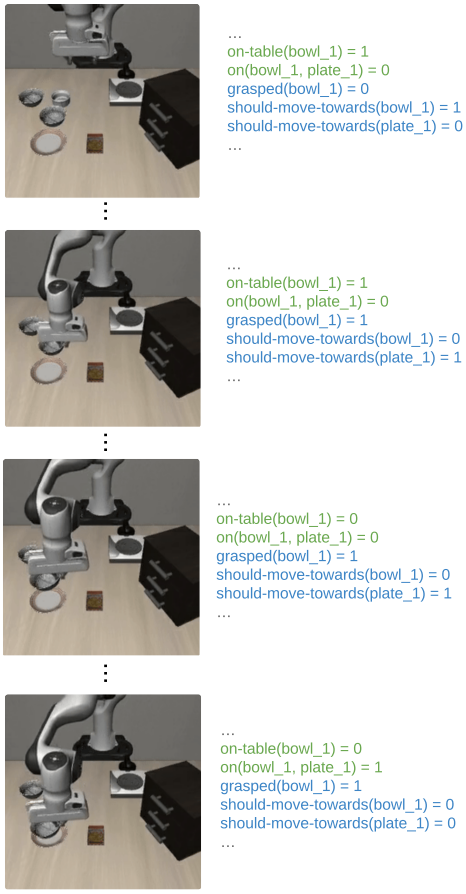

Fig. 3. **Example Labeled Object States and Action States in a LIBERO-spatial Pick-and-Place Trajectory.** Object states are shown in green and action states are shown in blue. The task is to "pick up the black bowl between the plate and the ramekin and place it on the plate". The first frame (top) is taken at the beginning of an episode where the target black bowl (which happens to be the *bowl_1* object) is still directly on the table and the robot needs to pick it up. As expected, the *on-table(bowl_1)* object relation and the *should-move-towards(bowl_1)* action subgoal are detected to be true. Once the robot arm closes its grippers on the target black bowl in frame 2, the *grasped(bowl_1)* action status is detected to be true. In frame 3, the *on-table(bowl_1)* object relation becomes false as the bowl is lifted off the tabletop. Finally, in frame 4, the *on(bowl_1, plate_1)* object relation becomes true, indicating that the task has been successfully completed.

IV. Probing Experiment

To test our hypothesis, we extract activations from the 33 hidden layers of OpenVLA's Llama 2 7B backbone. Each hidden-layer embedding is a 4096-dimensional vector. We then train two probes on each layer's activations to predict object states and action states. In total, we train $2 \times 33 = 66$ probes.

An object state involves the following relation predicates for the LIBERO-spatial suite: *behind(tabletop-object1, tabletop-object2)*, *in-front-of(tabletop-object1, tabletop-object2)*, *inside(tabletop-object, container)*, *left-of(tabletop-object1, tabletop-object2)*, *on(tabletop-object1, tabletop-object2)*, *on-table(tabletop-object)*, and *right-of(tabletop-object1, tabletop-object2)*, as well as unary object property predicates: *open(container)* and *turned-on(on-off-object)*.

An action state captures the action status predicate *grasped(pickupable-object)* and the action subgoal predicate *should-move-towards(tabletop-object)*. Examples of object states and action states are provided in the Probe Training Data Collection section below. We find combinations of grounded objects to which a predicate is applicable, and we define the object relation atoms as the object relation predicates applied to all combinations of their grounded objects. We define an object state as a complete truth assignment to the object relation atoms and object property atoms. Similarly, we define an action state as a complete truth assignment to the action status atoms and action subgoal atoms. In total, a LIEBRO-spatial object state has 224 atoms and an action state

has 12 atoms, a LIBERO-object object state has 461 atoms and an action state has 20 atoms.

A. Probe Training Data Collection

To train the probes, we need to collect (hidden layer activation, ground truth state) pairs as training data. To do this, we implement detector functions that detect the truth values for object relations, object properties, action statuses, and action subgoals in the 10 LIBERO-object tasks and the 10 LIBERO-spatial tasks. Figure 2 and Figure 3 demonstrate the changes in symbolic states in a LIEBRO-object trajectory and a LIBERO-spatial trajectory respectively. The 4 frames are points at which critical state changes happen.

We collect 5 successful episodes per LIBERO-object task and 5 per LIBERO-spatial task by repeatedly querying OpenVLA until 5 completed episodes are obtained. While the model is queried to predict the next robot action at each frame, we also extract and record the activations from each hidden layer $\ell \in \{0, \ldots, 32\}$. Importantly, at timestep t, we pair the hidden-layer embedding $\mathbf{h}_t$ with the ground-truth symbolic state $\mathbf{y}_t$ at the *same* time, ensuring no temporal mismatch (e.g., not using $t+1$ states to label time t). We then store each pair as $(\mathbf{h}_t, \text{object state}_t)$ or $(\mathbf{h}_t, \text{action state}_t)$.

B. Probe Training Data Preprocessing

To assess the linear decodability of symbolic states from OpenVLA's internal representations, we employ linear probes trained on per-timestep embeddings and corresponding ground-truth symbolic labels derived from the LIBERO-Object environment. The symbolic labels are encoded using three values: '1' indicating the state is True, '0' indicating False, and '-1' indicating the state is Not Applicable (N/A), typically due to the absence of relevant objects in the current task scenario. Rigorous preprocessing and a specific training protocol are essential for obtaining reliable and interpretable results.

1) Episode-Level Data Splitting: Our dataset consists of multiple trajectories (episodes) collected from interactions within the environment. To prevent temporal leakage, where information from adjacent frames within the same trajectory could inflate performance on held-out data, we perform data splitting strictly at the episode level *before* any label filtering or analysis. Each complete episode is randomly assigned to either a training set (90% of episodes) or a validation set (10% of episodes), using a fixed random seed for reproducibility. This ensures that the probe is trained on trajectories entirely disjoint from those used for validation, compelling it to generalize across different interaction scenarios rather than merely interpolating within a familiar trajectory. The validation set is used solely for reporting final performance metrics.

2) Label Filtering based on Training Set Frequencies: Many symbolic states may exhibit minimal or no variation within the collected data, offering little insight into the model's dynamic state representation. To focus the probing analysis on informative labels, we implement a filtering step based on label frequency, calculated **exclusively** using the designated **training set episodes** to avoid any data leakage from the validation set into the feature selection process.

Specifically, for each symbolic label dimension, we concatenate all corresponding label values ('-1', '0', or '1') from all timesteps within the training episodes. We then compute the frequency of the 'True' state ('1') relative to the 'False' state ('0'), explicitly **ignoring the 'Not Applicable' ('-1') entries** for this calculation. That is, the frequency f for label i is calculated as: $f_i = \sum(y_i == 1) / \sum(y_i != -1)$ across all training timesteps where y_i is the label value. Any label i where f_i is below 1% or above 99% is deemed near-constant *with respect to its applicable states* and is **excluded** from the set of labels (`keep_indices`) that the probes are trained to predict. This ensures the probes focus on labels demonstrating meaningful variation within the training data distribution. The same set of `keep_indices` derived from the training set is applied when preparing data for validation.

3) Masking Non-Applicable Labels during Training and Evaluation: A core design choice in this probing methodology is how to handle the '-1' (Not Applicable) labels. Instead of treating '-1' as a third class to be predicted, we **mask** these entries during both loss computation and metric calculation.

Motivation: Our primary goal is to determine if the *truth value* (True/False) of a symbolic state, *when it is relevant*, is linearly decodable from the embedding. Predicting object presence or state applicability is a distinct, albeit related, task. By masking '-1' values, we directly probe the representation of the 0/1 distinction without confounding the analysis with the model's potential ability (or inability) to infer applicability solely from the embedding. This is standard practice in probing when focusing on the semantics of defined states.

Mechanism: During the training and validation loop (`run` function):

1) The ground-truth label tensor `y` (containing '-1', '0', '1' for the kept labels) is loaded.
2) A boolean mask `mask = (y != -1)` is created.
3) The target for the Binary Cross-Entropy loss is defined as `target = (y == 1).float()`, effectively mapping '1' to '1.0' and both '0' and '-1' to '0.0'.
4) The BCE loss is computed element-wise but then multiplied by `mask.float()`. The final loss for the batch is the sum of the masked element-wise losses divided by the number of valid (non-masked) elements (`mask.sum()`). This ensures only timesteps where the label was '0' or '1' contribute to the loss and gradient updates.
5) Similarly, accuracy and F1-score calculations operate only on the masked elements. Predictions `pred` (thresholded sigmoid outputs, '0, 1') are compared against `target` only where `mask` is True (`pred[mask] == target[mask]`). The total number of valid predictions (`tot = mask.sum().item()`) serves as the denominator for accuracy. F1 scores are computed using only the masked predictions and targets.

4) Feature Usage and Class Balancing: We utilize the raw embeddings extracted directly from the specified OpenVLA

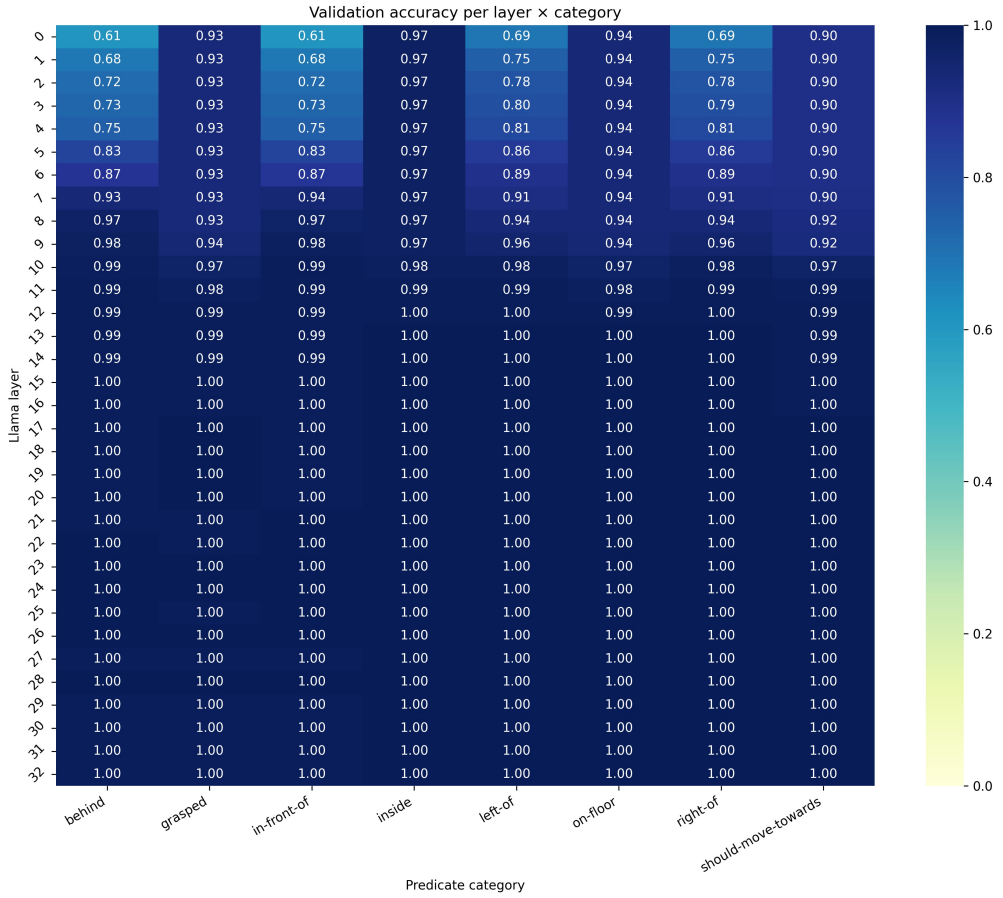

Fig. 4. **Probing Results of the LIEBRO-object dataset.** The first seven columns are object state symbols and the last two columns are action state symbols.

layers without applying z-score standardization or other normalization techniques, as preliminary experiments indicated stable convergence without them. Furthermore, no explicit class re-balancing techniques (e.g., over/under-sampling, loss weighting) are applied to the 0/1 labels after masking. While imbalance exists for certain labels (as shown in Figure X), the primary goal here is to assess linear separability in its raw form. The filtering step already removes the most extreme cases (labels that are almost always 0 or always 1 when applicable).

5) Summary of Preprocessing and Training: In summary, our probe training pipeline involves: (1) Strict episode-level splitting of data into training and validation sets. (2) Filtering of labels based on the frequency of True vs. False states (ignoring N/A states) calculated *only* on the training set. (3) Training linear probes using Binary Cross-Entropy loss where non-applicable ('-1') target labels are masked and do not contribute to the loss or gradients. (4) Evaluating probes using accuracy and Macro F1-score, similarly masking non-applicable labels to ensure metrics reflect performance solely on the True/False classification task for relevant states. This rigorous process prevents data leakage and focuses the analysis on the linear decodability of meaningful, dynamic symbolic states represented within OpenVLA.

C. Probe Training and Evaluation

Our probing methodology builds on recent work investigating internal representations in language models [16] and multimodal embeddings [18]. Inspired by these approaches, we implement a linear probe that maps from the model's internal representations to ground truth environment states. However, rather than using single-label classification which would face combinatorial explosion with growing numbers of states, we extend this to multi-label classification where each state variable can be predicted independently.

Formally, for a given layer's activation vector $\mathbf{h} \in \mathbb{R}^d$, our probe learns a mapping to binary predictions $\hat{\mathbf{y}} \in [0,1]^n$ where n is the number of tracked symbolic states:

$$\hat{\mathbf{y}} = \sigma(\mathbf{W}\mathbf{h} + \mathbf{b}) \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{n \times d}$ and $\mathbf{b} \in \mathbb{R}^n$ are learned parameters and σ is the sigmoid activation function. Each element of $\hat{\mathbf{y}}$ corresponds to a binary prediction about a specific ground atom (e.g., "on(bowl_1, plate_1)" or "in(bowl_1, top_drawer"). We train using binary cross-entropy loss with the Adam optimizer.

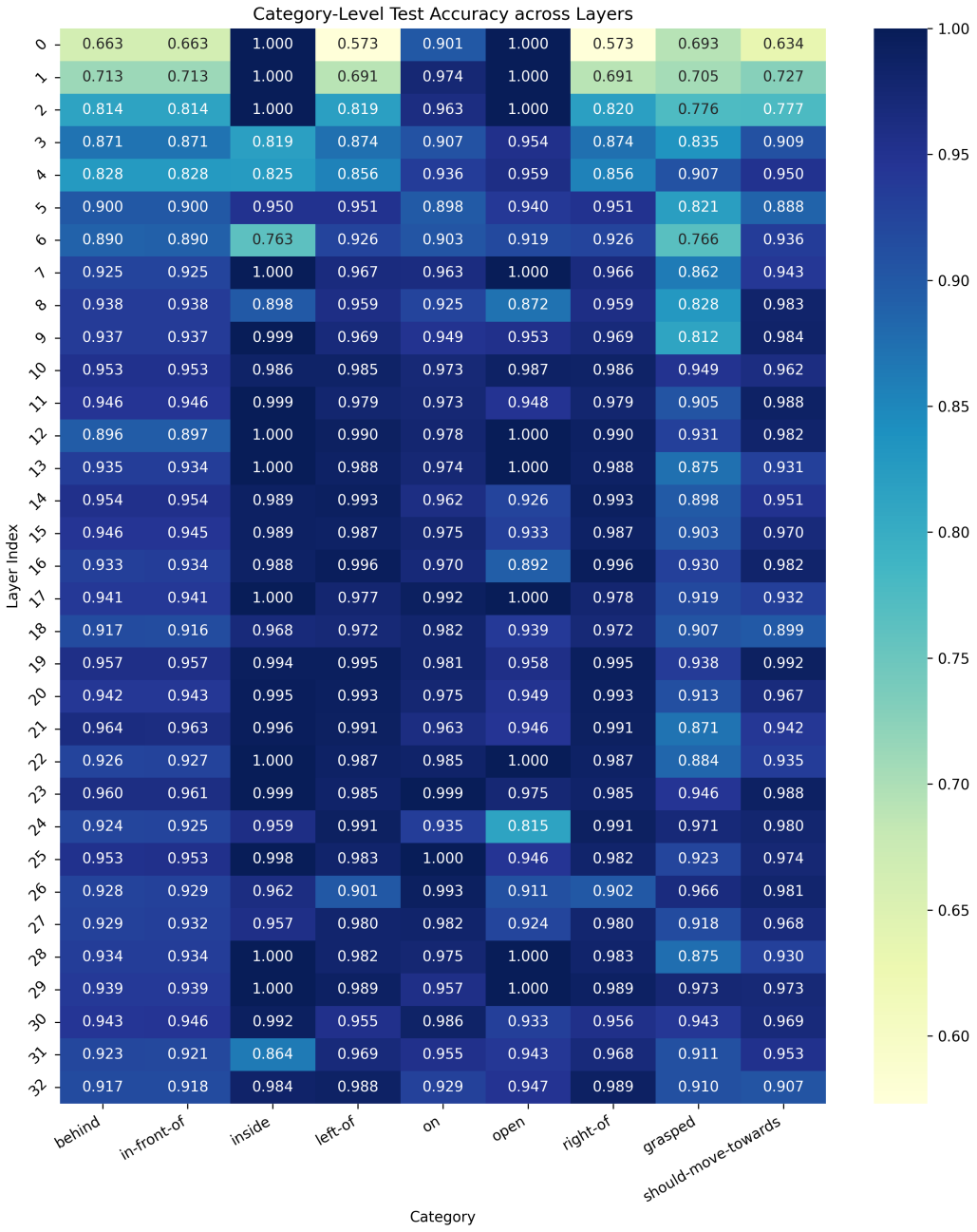

Fig. 5. **Probing Results of the LIEBRO-spatial dataset.** The first seven columns are object state symbols and the last two columns are action state symbols.

For evaluation, we compute averaged accuracies per predicate type. For a predicate like "on", we average the accuracies across all specific instances of that predicate that we track - for example, if we track both "on(bowl_1, plate_1)" and "on(plate_1, table_1)", we would average their individual prediction accuracies to get the overall accuracy for the "on" predicate:

$$\text{acc}(\text{pred}) = \frac{1}{N_{\text{pred}}} \sum_{i=1}^{N_{\text{pred}}} \text{acc}(i) \tag{2}$$

where N_{pred} is the number of tracked instances of that predicate, and $\text{acc}(i)$ is the binary prediction accuracy for the ith instance.

The probe results, visualized as a heatmap across layers and predicates, reveal consistently high accuracies across later layers suggesting robust encoding of symbolic state information. However, the first layer shows notably lower performance, aligning with its expected role in encoding lower-level features rather than high-level semantic relationships.

V. Results and Discussion

Figures 4 and 5 show a heatmaps of the object state probes' accuracies and the action state probes' accuracies across all 33 layers. Note that the object relation *on-table(tabletop-object)* and the object property *turned-on(on-off-object)* are dropped from training due to low variance in the training data-most objects remain on-table and the *stove_1* object (the only on-off-object) remains off, therefore their corresponding atoms never change truth values.

The accuracies are above 0.90 for most layers, indicating that the OpenVLA indeed encodes some object relation, object property, action status, and action subgoal features. The layer 0 probes perform significantly worse across all categories. This is not surprising as the first Llama layer probably only encodes very low-level semantic features such as syntactic relations and not the high level visual-semantic features such as object relations. We do not observe the hypothesized pattern of higher object state accuracies in earlier layer probes versus that of later layer probes. This does not support our hypothesis 1 and hypothesis 2. We recognize that the training data we use are not diverse enough in terms of the variation in object states and action states. Specifically, the objects in the 10 simulated LIBERO-spatial tasks have the same placements across tasks except for the two black bowls. As a result, most of the object relations remain unchanged across tasks, significantly reducing the variation in object states. Furthermore, the robot always picks one of the two black bowls, and the place target is always the plate, significantly reducing the variation in action states. These two factors combined significantly reduce the difficulty of the linear classification task that the probes are trained to do, leading to high accuracies across layers and categories, potentially washing out the layer-wise object state versus action state difference we expected to observe. More and better data are needed to test our hypotheses.

VI. Limitation and Future Work

Our current evaluation is limited due to the lack of diversity of the LIBERO-spatial tasks as object layouts remain relative unchanged across episodes. This potentially masks the differences across model layers as probe performance is inflated. As future work, we plan to scale the probing experiment up by collecting more diverse data from tasks that involve variable objects, variable object layouts, and variable goals.

The integration of symbolic probing and CA shows great potential for real-world robotic applications in which explainability and safety are critical. In these applications, real-time access to interpretable symbolic states allows the CA to monitor and potentially correct robot actions during execution. We believe that extracting symbolic information from VLAs opens the door to the integration of CA and VLA and we demonstrate an integrated CA-VLA system with this work. In the future, we hope to explore how the reasoning capabilities of the CA can enhance or monitor the performance of the VLA.

References

[1] O. M. Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, T. Kreiman, C. Xu, J. Luo, Y. L. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. [Online]. Available: https://arxiv.org/abs/2405.12213v2

[2] M. J. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. OpenVLA: An open-source vision-language-action model. [Online]. Available: https://arxiv.org/abs/2406.09246v3

[3] Z. Wang, Z. Zhou, J. Song, Y. Huang, Z. Shu, and L. Ma, "Towards testing and evaluating vision-language-action models for robotic manipulation: An empirical study." [Online]. Available: http://arxiv.org/abs/2409.12894

[4] V. N. Gudivada, "Chapter 1 - cognitive computing: Concepts, architectures, systems, and applications," in *Handbook of Statistics*, ser. Cognitive Computing: Theory and Applications, V. N. Gudivada, V. V. Raghavan, V. Govindaraju, and C. R. Rao, Eds. Elsevier, vol. 35, pp. 3–38. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0169716116300451

[5] M. Scheutz, T. Williams, E. Krause, B. Oosterveld, V. Sarathy, and T. Frasca, "An overview of the distributed integrated cognition affect and reflection DIARC architecture," in *Cognitive Architectures*, M. I. Aldinhas Ferreira, J. Silva Sequeira, and R. Ventura, Eds. Springer International Publishing, vol. 94, pp. 165–193, series Title: Intelligent Systems, Control and Automation: Science and Engineering. [Online]. Available: http://link.springer.com/10.1007/978-3-319-97550-4_11

[6] N. Chen, N. Wu, S. Liang, M. Gong, L. Shou, D. Zhang, and J. Li, "Is bigger and deeper always better? probing LLaMA across scales and layers." [Online]. Available: http://arxiv.org/abs/2312.04333

[7] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models." [Online]. Available: http://arxiv.org/abs/2307.09288

[8] S. Wu, A. Oltramari, J. Francis, C. L. Giles, and F. E. Ritter, "Cognitive LLMs: Towards integrating cognitive architectures and large language models for manufacturing decision-making." [Online]. Available: http://arxiv.org/abs/2408.09176

[9] G. Bajaj, K. Pearce, S. Kennedy, O. Larue, A. Hough, J. King, C. Myers, and S. Parthasarathy, "Generating chunks for cognitive architectures," vol. 2, no. 1, pp. 246–252, number: 1. [Online]. Available: https://ojs.aaai.org/index.php/AAAI-SS/article/view/27683

[10] J. R. Kirk, R. E. Wray, and J. E. Laird, "Exploiting language models as a source of knowledge for cognitive agents." [Online]. Available: http://arxiv.org/abs/2310.06846

[11] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. v. Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill, E. Brynjolfsson, S. Buch, D. Card, R. Castellon, N. Chatterji, A. Chen, K. Creel, J. Q. Davis, D. Demszky, C. Donahue, M. Doumbouya, E. Durmus, S. Ermon, J. Etchemendy, K. Ethayarajh, L. Fei-Fei, C. Finn, T. Gale, L. Gillespie, K. Goel, N. Goodman, S. Grossman, N. Guha, T. Hashimoto, P. Henderson, J. Hewitt, D. E. Ho, J. Hong, K. Hsu, J. Huang, T. Icard, S. Jain, D. Jurafsky, P. Kalluri, S. Karamcheti, G. Keeling, F. Khani, O. Khattab, P. W. Koh, M. Krass, R. Krishna, R. Kuditipudi, A. Kumar, F. Ladhak, M. Lee, T. Lee, J. Leskovec, I. Levent, X. L. Li, X. Li, T. Ma, A. Malik, C. D. Manning, S. Mirchandani, E. Mitchell, Z. Munyikwa, S. Nair, A. Narayan, D. Narayanan, B. Newman, A. Nie, J. C. Niebles, H. Nilforoshan, J. Nyarko, G. Ogut, L. Orr, I. Papadimitriou, J. S. Park, C. Piech, E. Portelance, C. Potts, A. Raghunathan, R. Reich, H. Ren, F. Rong, Y. Roohani, C. Ruiz, J. Ryan, C. Ré, D. Sadigh, S. Sagawa, K. Santhanam, A. Shih, K. Srinivasan, A. Tamkin, R. Taori, A. W. Thomas, F. Tramèr, R. E. Wang, W. Wang, B. Wu, J. Wu, Y. Wu,

S. M. Xie, M. Yasunaga, J. You, M. Zaharia, M. Zhang, T. Zhang, X. Zhang, Y. Zhang, L. Zheng, K. Zhou, and P. Liang, "On the opportunities and risks of foundation models." [Online]. Available: http://arxiv.org/abs/2108.07258

[12] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, and S. Riedel, "Language models as knowledge bases?" [Online]. Available: http://arxiv.org/abs/1909.01066

[13] D. Alivanistos, S. B. Santamaría, M. Cochez, J.-C. Kalo, E. v. Krieken, and T. Thanapalasingam, "Prompting as probing: Using language models for knowledge base construction." [Online]. Available: http://arxiv.org/abs/2208.11057

[14] Z. Wang, L. Ye, H. Wang, W.-C. Kwan, D. Ho, and K.-F. Wong, "ReadPrompt: A readable prompting method for reliable knowledge probing," in *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics, pp. 7468–7479. [Online]. Available: https://aclanthology.org/2023.findings-emnlp.501

[15] S. Qi, Z. Cao, J. Rao, L. Wang, J. Xiao, and X. Wang, "What is the limitation of multimodal LLMs? a deeper look into multimodal LLMs through prompt probing," vol. 60, no. 6, p. 103510. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0306457323002479

[16] B. Z. Li, M. Nye, and J. Andreas, "Implicit representations of meaning in neural language models," in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Association for Computational Linguistics, pp. 1813–1827. [Online]. Available: https://aclanthology.org/2021.acl-long.143

[17] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "LIBERO: Benchmarking knowledge transfer for lifelong robot learning." [Online]. Available: http://arxiv.org/abs/2306.03310

[18] A. D. Lindström, S. Bensch, J. Björklund, and F. Drewes, "Probing multimodal embeddings for linguistic properties: the visual-semantic case," in *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 730–744. [Online]. Available: http://arxiv.org/abs/2102.11115

APPENDIX

A. The DIARC - OpenVLA - Probe Integration

a) WebSocket Server and Real-Time Flow.: We implement a lightweight WebSocket server to provide real-time communication among OpenVLA, the LIBERO simulator, DIARC, and a React UI:

1) **User Task.** The user selects a pick-and-place command in DIARC's GUI (e.g., "pick up the black bowl between the plate and the ramekin..."). DIARC sends this instruction via WebSocket to our server.
2) **Environment Step.** The server runs the LIBERO-spatial simulator, retrieving the latest camera frame for input to OpenVLA's policy. OpenVLA returns a 7D action, which the simulator executes.
3) **Probe Inference.** Simultaneously, the server extracts the hidden-layer embedding from OpenVLA, feeds it to our linear probes, and obtains predicted symbolic states (object relations, subgoals, etc.).
4) **Streaming Back.** Finally, the server encodes the current camera image in base64 and bundles it with the predicted symbolic states plus the timestep index. This data is streamed back over the WebSocket to DIARC and the React UI.

b) React UI and Timeline Scrubbing.: Figure 6 shows our React-based interface. While the task runs, the UI continuously displays:

- A *live camera feed* (at $\sim$5 Hz) pinned at the top-left, showing the robot's current manipulation.
- A *symbolic states* panel on the right, color-coding newly activated or deactivated predicates (e.g., green if *on-table(bowl_1)* flips from 0 to 1).
- A *timeline slider* becomes available once the task completes, letting the user "scrub" back through each timestep's image and states to analyze the model's evolution over time.

This design allows operators or domain experts to confirm that the predicted states mirror actual environment changes (e.g., verifying *on(bowl,plate)=1* precisely when the bowl is placed). Meanwhile, DIARC receives the same 0/1 states as symbolic predicates, enabling high-level logic or safety checks without manually altering the VLA policy. By decoupling the raw policy (OpenVLA) from DIARC's symbolic reasoning through a WebSocket-based design, we maintain modularity while allowing step-by-step monitoring of the policy's internal states. We thus leverage the *generalist* capabilities of a VLA model and the *reliability* of a symbolic architecture. This opens the door for future enhancements where DIARC might intervene upon contradictory states (e.g., an object can't be both "on the plate" and "in the cabinet") or respond to user queries mid-task. In short, our integration unites robust low-level action generation with high-level interpretability and control.

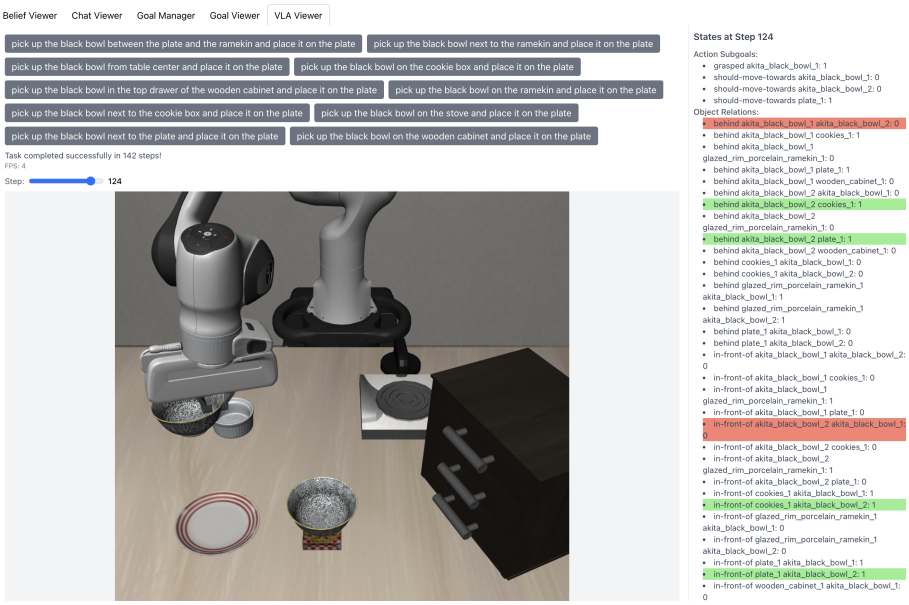

Fig. 6. **DIARC–OpenVLA GUI.** The left-hand pane displays the real-time camera feed (updated at 5–10 Hz), showing the robot's manipulation progress. The right-hand pane color-codes each predicted symbolic state (green for newly activated, red for deactivated), letting users quickly verify whether OpenVLA's internal representation matches the environment. After task completion, a timeline slider appears, allowing the user to revisit earlier steps' images and states for deeper analysis.