SEMAE: SEMANTIC-ALIGNED ACTION EXPERT FOR LANGUAGE-FOLLOWING ROBOT CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Attaching a new action expert with pretrained VLM to generate robot actions is the main paradigm for developing vision language action models. However, many experimental results have revealed that existing action experts significantly degrade the semantic knowledge contained in pretrained VLMs, leading to VLAs that, despite excelling at complex and dexterous manipulation, often fail to follow language instructions accurately. To address this challenge, we propose SemAE, an encoder-decoder action expert that enables mutual alignment between robot actions and pretrained VLM's semantic embeddings, allowing VLAs to seamlessly inherit the knowledge in VLMs. Specifically, the encoder introduces a motion primitive distillation technique to compress raw robot actions into a compact set of latent embeddings, which are aligned with the semantic space of the pretrained VLM through layer-wise motion primitive distillation. Moreover, we propose a patch-wise action flow decoder, which efficiently decompresses a single semantic embedding into future action sequence. Built on SemAE, SemVLA achieves strong language-following with fast inference via parallel decoding of semantic action embeddings. Extensive experiments across multiple standard benchmarks and real robots demonstrate that SemAE effectively aligns robot actions with semantic knowledge, substantially enhancing the language-following capabilities of Vision-Language-Action models.

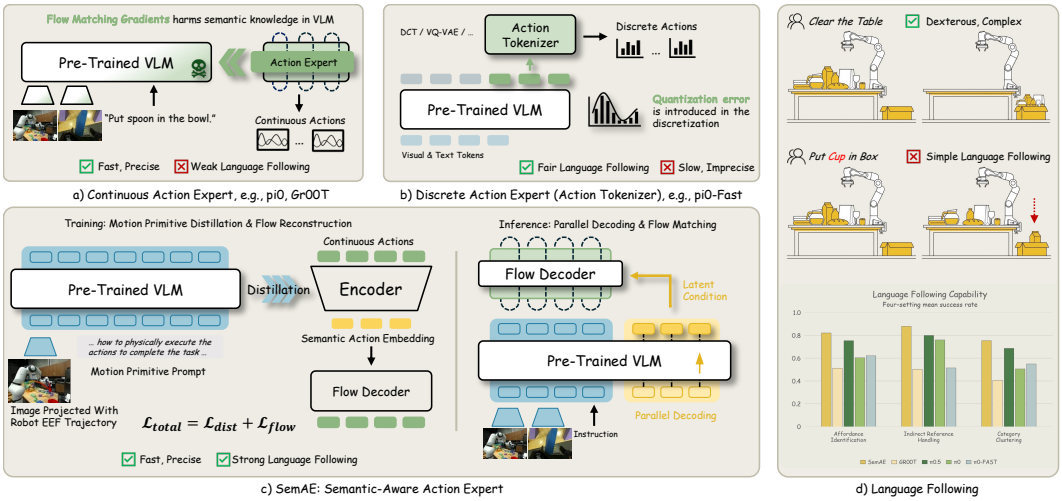


Figure 1: **Action experts comparisons. (a)** Widely used flow-matching experts like π_0 are weak at language following due to the degradation of VLM's semantic knowledge caused by flow-matching gradients (Driess et al., 2025). **(b)** Action tokenizers like π_0-FAST have better language-following capabilities but perform poorly on dexterous tasks due to quantization error. **(c)** Benefiting from motion primitive distillation and flow reconstruction, SemAE generates precise actions rapidly and follows language instructions well. **(d)** Traditional action experts can finish complex tasks such as clearing all items from the table, but they always struggle with simple language-following tasks, such as grasping a specified object from multiple manipulable items (Driess et al., 2025). Across three types of tasks (affordance identification, indirect reference, category clustering) used to evaluate model's language-following capability, SemAE consistently outperforms traditional action experts.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Vision-Language-Action (VLA) models have become a compelling paradigm for general-purpose robot control by scaling imitation learning to large robot datasets and performing diverse, dexterous skills such as folding laundry and making coffee. Nevertheless, their ability to follow language instructions remains limited, particularly in scenes with multiple feasible interaction targets, where the policy may execute an unintended variant of the command like picking the wrong object (Black et al., 2024; Pertsch et al., 2025; Intelligence et al., 2025). VLAs are commonly built by attaching a pretrained vision-language model (VLM) with an action expert, which serves as a dedicated module for action modeling (e.g., continuous flow-matching/diffusion experts or discrete action tokenizers), aiming to inherit semantic knowledge from web-scale VLM pretraining while still producing precise control signals. However, recent studies suggest that naively backpropagating gradients from a newly introduced action expert, whose parameters are typically initialized from scratch, into the pretrained VLM backbone can interfere with the semantic knowledge in the pretrained VLM, ultimately degrading language following even when manipulation proficiency improves (Black et al., 2024; Intelligence et al., 2025). These observations highlight a core challenge: **how can we optimize the design of action experts to build VLAs with stronger language-following ability?**

We argue that progress on this question hinges on two coupled bottlenecks. First, when jointly training a pretrained VLM with a randomly initialized action expert, the gradients from the untrained expert can dominate the learning signal and corrupt the backbone's semantic representations, leading to a drop in instruction-following even if the policy becomes more proficient at low-level control. A natural workaround is restricting gradient flow from the expert into the backbone and co-training with general vision-language data. Yet such recipes typically rely on additional data beyond standard robot demonstrations, and when the model is trained on pure robot action data, the language-following ability still deteriorates over continued fine-tuning (Intelligence et al., 2025). Second, even if semantic knowledge is preserved, VLAs still need an interaction interface between the VLM and the action expert that transmits semantic knowledge without introducing a new bottleneck. Autoregressive VLAs address alignment by representing actions as discrete tokens, but they inherit the sequential decoding latency of language models and can suffer from quantization errors that hurt dexterous control (Pertsch et al., 2025; Wang et al., 2025). These challenges call for an action expert whose latent space is naturally aligned with the backbone's semantics, and a VLM-expert interaction pattern that is both semantically expressive and fast enough for real-time control.

In this work, as shown in Figure 1(c), we propose the Semantic-Aligned Action Expert, **SemAE**, and a SemAE-based general robot policy, **SemVLA**, to strengthen language following in continuous robot control. SemAE is an encoder-decoder action expert co-trained on robot demonstrations with two objectives: continuous action reconstruction and semantic distillation from a pretrained VLM. To align the action latent space with the VLM semantic space, we introduce ***layer-wise motion primitive distillation***, which matches the intermediate representations of the action encoder to the corresponding hidden states of a frozen VLM. This encourages the encoder to compress the raw robot action into a compact set of *semantic action embeddings*. Moreover, we design a ***patch-wise action flow decoder*** that decodes action semantic embedding into a continuous action patch via conditional rectified flow, enabling accurate and efficient generation of high-dimensional actions. Built on SemAE, SemVLA connects the VLM backbone to the action decoder through ***semantic parallel decoding***: we prefill a set of dedicated special tokens in the VLM input sequence, so a single forward pass produces multiple semantic action embeddings that are directly consumed by SemAE for fast action decoding. Our contributions are summarized as follows:

- We propose **SemAE**, an encoder-decoder action expert that aligns continuous robot actions with a pretrained VLM semantic space through layer-wise motion primitive distillation.
- We introduce **SemVLA**, a VLA that uses semantic parallel decoding to predict semantic action embeddings and decodes them with SemAE for fast inference.
- We develop three complementary language-following metrics, as shown in Figure 1(d), and improve over $\pi_{0.5}$ by 6.75, 8.00, and 6.75 percentage points on Affordance Identification, Indirect Reference Handling, and Category Clustering, respectively.

2 Related Work

Vision-Language-Action models. Vision-Language-Action models aim to build general robot policies that execute tasks from natural language instructions. RT-2 (Brohan et al., 2023) fine-tunes PaLI-X (Chen et al., 2024) with discretized action tokens, demonstrating that transferring semantic knowledge from web-scale pretraining can improve language following in robotic control, while OpenVLA (Kim et al., 2024) adapts Prismatic VLM (Karamcheti et al., 2024) on the OXE dataset (Collaboration et al., 2024), showing the ability to scale to diverse embodiments. More recently, π_0 (Black et al., 2024) shows that a VLA can execute a wide range of dexterous skills directly from human language prompts. $\pi_{0.5}$ (Intelligence et al., 2025) extends this direction toward higher-level instruction following by reasoning over a high-level command, decomposing it into language sub-instructions, and then executing each sub-instruction through low-level robot control.

Action expert. Many VLAs augment the pretrained backbone with an action expert, a dedicated module for action modeling that interfaces between semantic representations and control outputs. As a representative continuous expert, π_0 (Black et al., 2024) equips a pretrained VLM with a flow-matching action expert that generates continuous action chunks for fast, high-frequency control. KI finds that naively training such experts can interfere with pretrained semantics and harm language following, and it motivates insulating the backbone during adaptation (Driess et al., 2025). Discrete action experts instead convert action chunks into discrete tokens and decode them back to continuous trajectories. FAST (Pertsch et al., 2025) tokenizes actions by applying per-dimension DCT to an action chunk, quantizing the coefficients, and compressing the sparse integer sequence with BPE, with a low-frequency-first ordering that stabilizes autoregressive prediction. Learned VQ tokenizers instead train an encoder-decoder to discretize action sequences into codebook indices via residual quantization, as in VQ-VLA (Wang et al., 2025) and FASTer (Liu et al., 2025). VQ-VLA trains a convolutional residual VQ-VAE at scale and freezes it as an action tokenizer for an OpenVLA-style backbone (Wang et al., 2025). FASTer uses an action patchifier and an RVQ tokenizer that yields structured, coarse-to-fine codes for more efficient sequence modeling (Liu et al., 2025).

Unified tokenizers for conditional generation. UniFlow (Yue et al., 2026) studies unified visual tokenizers that support both visual understanding and conditional generation. It identifies a tension between preserving high-level semantic features and reconstructing fine-grained visual details. UniFlow addresses this by distilling semantic representations layer by layer from a pretrained encoder and by using a lightweight patch-wise pixel flow decoder conditioned on semantic features. SemAE applies a related design to robot action modeling: an encoder learns semantic-aligned action embeddings through VLM distillation, while a flow decoder reconstructs continuous actions. These embeddings then serve as prediction targets for the VLM, providing a compact semantic interface between robot observations, language instructions, and continuous action generation.

3 Method

Training has two stages: autoencoder training and VLM training, both using the same robot demonstration data. In Stage 1 (Section 3.1), we freeze the VLM and optimize the autoencoder with reconstruction and distillation losses simultaneously. We project robot trajectories onto observation images and feed the resulting images to the VLM with language instructions to produce language embeddings containing motion primitives at each transformer layer. A trainable Q-Former extracts these layer-wise embeddings and matches their sequence lengths to those of the encoder features, providing distillation targets for corresponding encoder layers. It is discarded after Stage 1. Meanwhile, the encoder compresses robot actions into semantic-aligned action embeddings in the VLM's semantic space, and the flow decoder reconstructs continuous actions from them. In Stage 2 (Section 3.2), we freeze the encoder and train the VLM to predict semantic-aligned action embeddings using mean squared error (MSE) loss. Under the standard training paradigm, the VLM takes observation images and text along with the pre-filled query token $\langle \texttt{ACT} \rangle$ (the first blue block and the yellow block in Figure 3). After a single forward pass, the final-layer embedding at the query-token position is compared with the frozen encoder's output to compute the MSE loss. For text-action co-training, the query token is embedded in a multi-turn text sequence, and the VLM is jointly optimized with MSE and text cross-entropy (CE) losses. During inference, only the VLM and decoder are needed. The VLM predicts semantic-aligned action embeddings from robot observations and language instructions, and the decoder converts them into continuous robot actions.

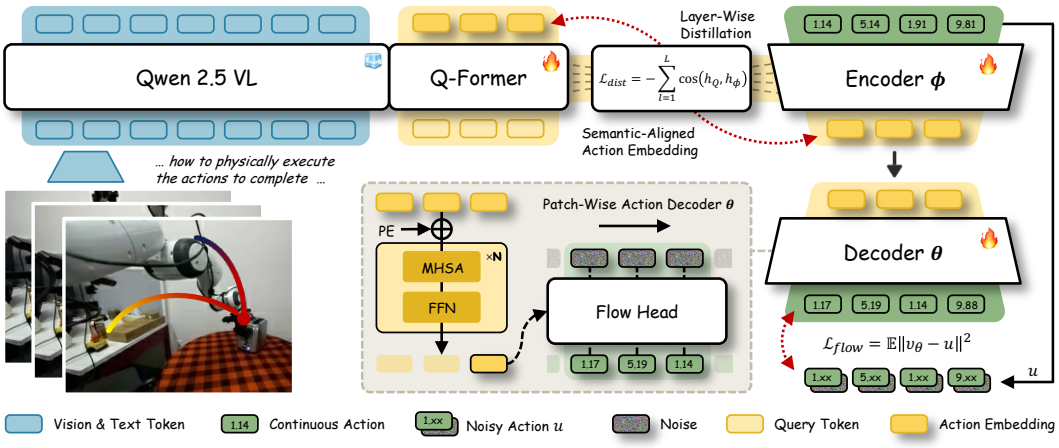


Figure 2: **Training pipeline of SemAE.** SemAE is an encoder-decoder action expert. During training, a frozen VLM takes text and images with projected action trajectories as input to generate language embeddings with motion primitives at each transformer layer. The encoder ϕ compresses continuous actions into semantic-aligned action embeddings through layer-wise distillation from the frozen VLM. To ensure action embeddings at each encoder layer are supervised by the corresponding language embeddings, a trainable Q-Former is attached to the VLM to align their sequence lengths. Conditioned on semantic-aligned action embeddings compressed by the encoder ϕ, the patch-wise action decoder θ reconstructs continuous actions from noise with a flow head.

3.1 SemAE: Semantic-Aligned Action Expert

As shown in Figure 2, SemAE is an encoder-decoder action expert that transfers semantic knowledge between continuous robot actions and VLM's semantic space. The encoder uses multi-layer 1D convolutions followed by a lightweight Transformer. The convolutions extract action motion primitives, and the Transformer produces semantic-aligned action features. The decoder is a lightweight Transformer-based rectified flow decoder. Conditioned on semantic embeddings from the VLM, it predicts a velocity field in action space and outputs continuous action chunks. We jointly train the encoder and decoder with action reconstruction and VLM distillation losses, as detailed below.

Layer-wise Motion Primitive Distillation. Prior work shows that projecting the end-effector (EEF) trajectory in 3D space onto a robot camera image provides an effective visual abstraction of high-dimensional continuous actions, and it helps VLMs understand robot motion in a semantically grounded way (Gu et al., 2024). Following this insight, we distill motion-related semantics from a frozen Qwen2.5VL teacher into the SemAE action encoder ϕ. Concretely, at time step t, we build a trajectory overlay image $\tilde{\mathbf{x}}_t$ by projecting the EEF positions over the next 2 seconds onto the robot camera image $\mathbf{x}_t$. The teacher takes $\tilde{\mathbf{x}}_t$ together with a system prompt that specifies the control convention, and produces layer-wise hidden states $\{\mathbf{H}_{\mathrm{T}}^{(l)}\}_{l=1}^{L}$. Not all teacher features are equally relevant to motion. We therefore introduce a Q-Former to extract motion-related embeddings from every teacher layer. Given the teacher hidden states at layer l, the Q-Former outputs a fixed set of S motion embeddings $\mathbf{M}^{(l)} \in \mathbb{R}^{S\times D}$. We use $\{\mathbf{M}^{(l)}\}_{l=1}^{L}$ as the distillation targets. On the student side, the action encoder ϕ maps an action chunk $\mathbf{a}_{t:t+\Delta}$ over the same 2-second horizon into intermediate hidden states $\{\mathbf{H}_{\phi}^{(l)}\}_{l=1}^{L}$, where $\mathbf{H}_{\phi}^{(l)} \in \mathbb{R}^{S\times D}$ denotes the S semantic action embeddings at layer l. To preserve high-level semantics more strongly while allowing low-level layers to stay flexible, we first define a hierarchical prior $w_l^{\text{base}} = \frac{l}{L}$ for each layer l. We then compute an alignment penalty α_l as the average cosine distance between student semantic action embeddings and teacher motion embeddings, $\alpha_l = 1 - \frac{1}{S}\sum_{i=1}^{S} \frac{\langle \mathbf{H}_{\phi}^{(l,i)}, \mathbf{M}^{(l,i)} \rangle}{\|\mathbf{H}_{\phi}^{(l,i)}\| \, \|\mathbf{M}^{(l,i)}\|}$. The final layer-wise weight w_l is a normalized combination of these two factors:

$$w_l = \frac{w_l^{\text{base}} \cdot \exp(\beta \cdot \alpha_l)}{\sum_{k=1}^{L} w_k^{\text{base}} \cdot \exp(\beta \cdot \alpha_k)}, \tag{1}$$

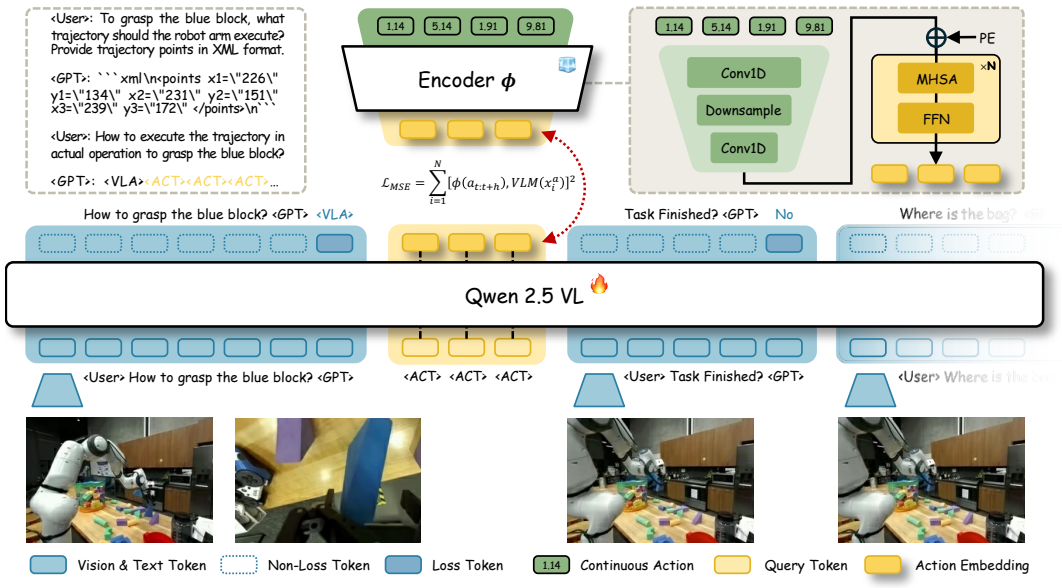


Figure 3: **Training pipeline of SemVLA.** The frozen encoder trained in Figure 2 is used to supervise VLM training by producing semantic-aligned action embeddings. The special query token $\langle \texttt{ACT} \rangle$ is fed to the trainable VLM. After one forward pass, the final-layer embedding is extracted to compute the MSE loss against the action embeddings generated by the encoder. Furthermore, this training paradigm is naturally suited for text-action co-training. The query token $\langle \texttt{ACT} \rangle$ can be embedded in multi-turn text, so the VLM is optimized with action and text losses simultaneously, further preserving its semantic knowledge. Since action embeddings are aligned with text embeddings in the same semantic space, SemAE can further improve the performance of text-action co-training.

where β is a temperature hyperparameter that emphasizes poorly aligned layers. Finally, the layer-wise motion primitive distillation loss is the weighted sum of cosine distances across layers:

$$\mathcal{L}_{\text{dist}} = \sum_{l=1}^{L} w_l \cdot \left(1 - \frac{1}{S} \sum_{i=1}^{S} \frac{\langle \mathbf{H}_{\phi}^{(l,i)}, \mathbf{M}^{(l,i)} \rangle}{\|\mathbf{H}_{\phi}^{(l,i)}\| \, \|\mathbf{M}^{(l,i)}\|} \right). \tag{2}$$

This objective transfers hierarchical motion semantics into ϕ, encouraging convolutions to form motion primitives and Transformer layers to progressively align them with VLM semantics.

Patch-wise Action Flow Decoding. The decoder learns to reconstruct continuous action chunks from semantic action embeddings using latent-conditioned rectified flow (Liu et al., 2022; Lipman et al., 2022). For ground-truth action chunk $\mathbf{a}_{t:t+\Delta} \in \mathbb{R}^{T \times d_a}$, we partition it into S non-overlapping action patches $\{\mathbf{a}^{(i)}\}_{i=1}^{S}$, where $\mathbf{a}^{(i)} \in \mathbb{R}^{p \times d_a}$ and $T = S \cdot p$. The action encoder ϕ produces S semantic action embeddings at its last layer, denoted as $\mathbf{H}_{\phi}^{(L)} = \{\mathbf{c}^{(i)}\}_{i=1}^{S}$ with $\mathbf{c}^{(i)} \in \mathbb{R}^{D}$. These embeddings serve as patch-wise conditions for the flow decoder v_θ. During SemAE training, $\mathbf{c}^{(i)}$ comes from $\phi(\mathbf{a}_{t:t+\Delta})$. When SemAE is used inside SemVLA, $\mathbf{c}^{(i)}$ is predicted by the VLM and directly decoded into action patches. For each patch $\mathbf{a}^{(i)}$, we sample a flow time $s \sim p_s$ and Gaussian noise $\boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(0, I)$, and define a linear interpolation trajectory:

$$\mathbf{a}_s^{(i)} = (1 - s)\mathbf{a}^{(i)} + s \cdot \boldsymbol{\epsilon}^{(i)}, \quad s \in [0, 1]. \tag{3}$$

This trajectory has a constant velocity $\mathbf{u}^{(i)} = \boldsymbol{\epsilon}^{(i)} - \mathbf{a}^{(i)}$. The decoder is trained to predict this velocity from the noisy patch $\mathbf{a}_s^{(i)}$, the time s, and its semantic condition $\mathbf{c}^{(i)}$. We train the decoder with the standard flow matching loss, averaged over all action patches in each chunk:

$$\mathcal{L}_{\text{flow}} = \mathbb{E}_{\mathbf{a}^{(i)} \sim p_{\text{data}},\ \boldsymbol{\epsilon}^{(i)} \sim \mathcal{N}(0,I),\ s \sim p_s} \left\| v_\theta(\mathbf{a}_s^{(i)}, s, \mathbf{c}^{(i)}) - (\boldsymbol{\epsilon}^{(i)} - \mathbf{a}^{(i)}) \right\|_2^2. \tag{4}$$

Together with the motion primitive distillation loss in Equation (2), this objective trains SemAE to compress actions into semantic action embeddings and decode them into accurate control.

3.2 SEMVLA: VLA WITH STRONGER LANGUAGE FOLLOWING

SemVLA uses a Qwen2.5VL backbone and the SemAE decoder (Figure 3). We freeze the SemAE encoder and train only the VLM to predict semantic-aligned action embeddings.

Semantic Parallel Decoding. The VLM communicates with SemAE through dedicated semantic action slots to transfer semantic knowledge to the action expert. Given the observation image $\mathbf{x}_t$ and the language instruction, we append S special tokens $\langle \texttt{ACT} \rangle$ to the VLM input sequence, where each token corresponds to one action patch in SemAE. These tokens provide a fixed interface in the VLM's semantic space for representing a whole action chunk. We initialize the action slots with learned embeddings and use bidirectional attention in the language decoder, so that every action slot can attend to the full multimodal context and to other slots. After a single forward pass, we take the final-layer hidden states at the S action slots and map them with a lightweight prediction head into S semantic action embeddings $\hat{\mathbf{c}}^{(i)} \in \mathbb{R}^D$. We then feed $\{\hat{\mathbf{c}}^{(i)}\}_{i=1}^S$ into the SemAE flow decoder v_θ to generate S action patches and form the continuous action chunk $\hat{\mathbf{a}}_{t:t+\Delta}$.

During training, we supervise these predicted semantic embeddings using the frozen SemAE encoder ϕ. For each demonstration action chunk $\mathbf{a}_{t:t+\Delta}$, the encoder produces semantic targets $\{\mathbf{c}^{(i)}\}_{i=1}^S = \phi(\mathbf{a}_{t:t+\Delta})$ as in Section 3.1. We optimize the VLM by minimizing the mean squared error between its predictions and the semantic-aligned action embeddings from the frozen encoder:

$$\mathcal{L}_{\text{spd}} = \frac{1}{S} \sum_{i=1}^{S} \left\| \hat{\mathbf{c}}^{(i)} - \text{sg}\Big(\mathbf{c}^{(i)}\Big) \right\|_2^2, \tag{5}$$

where $\text{sg}(\cdot)$ denotes the stop-gradient operator.

Scalability in Text-Action Co-Training. SemAE aligns action embeddings with text embeddings in the VLM's semantic space, making it naturally suited for text-action co-training. Compared with conventional action experts without explicit semantic alignment, this shared space allows text supervision to more directly support action prediction. As shown in Figure 3, we interleave text QA and action prediction in multi-turn sequences, following EO-1 (Qu et al., 2025a), and mix these sequences with robot demonstrations. At action turns, we insert S special query tokens $\langle \texttt{ACT} \rangle$. The VLM predicts semantic-aligned action embeddings from their final-layer states, using the frozen encoder ϕ to provide action targets. We jointly optimize text QA and action prediction with

$$\mathcal{L}_{\text{co}} = \mathcal{L}_{\text{QA}}^{\text{CE}} + \mathcal{L}_{\text{spd}}, \tag{6}$$

where $\mathcal{L}_{\text{QA}}^{\text{CE}}$ is the next-token cross-entropy loss on text answers, and $\mathcal{L}_{\text{spd}}$ is the action-embedding MSE loss defined in Equation (5). Text QA helps preserve the VLM's semantic knowledge while the model learns to predict semantic-aligned action embeddings within the same semantic space.

4 EXPERIMENTS

Our experiments assess whether aligning robot actions with pretrained VLM semantics improves both manipulation and language following. We address four questions:

1. How well does SemVLA perform on basic manipulation tasks?
2. Does SemAE improve a VLA's language-following ability?
3. How do the key design choices affect performance?
4. Can additional text-action co-training further improve performance?

Evaluation setup. We evaluate SemAE on four simulation benchmarks—LIBERO, LIBERO-PRO, SimplerEnv, and Polaris—and three real-world settings: Franka, WidowX 250s, and DROID. Franka covers object selection, grasping, placement, and multi-stage tasks; WidowX 250s tests instruction-conditioned manipulation with object, attribute, and spatial distractors. DROID includes cluttered kitchen and dining-table tasks, plus color-based cube placement. Figure 4 illustrates the simulation, Franka, and WidowX setups. We report task success over repeated rollouts, with intermediate grasp success where applicable. Benchmark and baseline descriptions appear in Appendices A.10.1 and A.10.2, and real-world protocols in Appendix A.10.3. Implementation details are provided in Appendix A.8.

Table 1: **Real-world WidowX results.** We compare task success and intermediate grasp success rates (%) across six task groups, covering object selection, placement, and microwave closing.

Model	Put Carrot on Plate		Put Eggplant in Basket		Put Purple Cup on White Plate		Put Green Cup on Pink Cloth		Close Microwave	Lift Red Chili Pepper	#Overall
	Grasp Carrot	Success	Grasp Eggplant	Success	Grasp Purple Cup	Success	Grasp Green Cup	Success	Success	Success	Average
OCTO (Octo Model Team et al., 2024)	22.7%	4.5%	36.4%	9.1%	0.0%	0.0%	18.2%	13.6%	54.5%	45.5%	21.2%
OpenVLA (Kim et al., 2024)	63.6%	59.1%	63.6%	63.6%	63.6%	18.2%	47.7%	40.9%	**100.0%**	72.7%	59.1%
OpenVLA-OFT (Kim et al., 2025)	68.2%	68.2%	72.7%	63.6%	54.5%	36.4%	52.3%	47.7%	**100.0%**	54.5%	61.7%
CogAct (Li et al., 2024a)	63.6%	59.1%	63.6%	45.5%	45.5%	36.4%	45.5%	40.9%	90.9%	63.6%	56.1%
SpatialVLA (Qu et al., 2025b)	81.8%	68.2%	81.8%	72.7%	54.5%	45.5%	84.1%	81.8%	**100.0%**	72.7%	73.5%
π_0 (Black et al., 2024)	77.3%	72.7%	81.8%	81.8%	63.6%	63.6%	79.5%	79.5%	**100.0%**	72.7%	78.4%
π_0-FAST (Pertsch et al., 2025)	31.8%	27.3%	27.3%	18.2%	36.4%	36.4%	25.0%	18.2%	45.5%	27.3%	28.8%
GR00T N1 (Bjorck et al., 2025)	31.8%	27.3%	36.4%	27.3%	27.3%	18.2%	22.7%	20.5%	63.6%	36.4%	32.2%
$\pi_{0.5}$ (Intelligence et al., 2025)	81.8%	77.3%	81.8%	81.8%	**81.8%**	**72.7%**	90.9%	81.8%	**100.0%**	90.9%	84.1%
SemAE	**90.9%**	**81.8%**	**100.0%**	**90.9%**	**81.8%**	**72.7%**	**95.5%**	**90.9%**	**100.0%**	**100.0%**	**89.4%**

Language-following metrics. We classify language-following tasks by the semantic information needed to identify the intended target. **Affordance Identification** tests whether a policy selects the instructed object among multiple manipulable candidates using explicit attributes, such as choosing a blue can rather than a red one. **Indirect Reference Handling** tests whether a policy resolves an object from relational or functional descriptions instead of its explicit name, such as "the item left of the cup" or "the tool used for cutting." **Category Clustering** tests whether a policy grounds a category-level instruction in its member objects, such as selecting food while ignoring non-food objects. Each metric is evaluated through the success rates of the corresponding tasks, rather than a separate language-only test.

Aggregation across platforms. For each metric, we evaluate the corresponding tasks in four settings: **Simulation, Franka, WidowX 250s, and DROID**. Let $s_{m,p,c}$ denote the success rate of model m on tasks assigned to category c in setting p. The aggregate score in Figure 1(d) gives each setting equal weight:

$$S_{m,c} = \frac{1}{4} \sum_{p=1}^{4} s_{m,p,c}. \tag{7}$$

Thus, each of the three scores averages four platform-specific success rates, rather than pooling all trials across platforms. The first three panels of Figure 5 show this breakdown; the fourth reports general manipulation performance separately. Task-mapping examples are provided in Appendix A.1; detailed task protocols appear in Appendices A.10.1 and A.10.3.



Figure 4: **Simulation and real-world evaluation setups.** From left to right: WidowX 250s and Franka robot platforms, followed by Polaris, SimplerEnv, and LIBERO/LIBERO-PRO simulation environments. The panels show robot hardware, camera placement, and manipulation scenes.

4.1 Basic Manipulation Performance

Simulation. SemAE achieves the highest average success rates on LIBERO (98.1%), LIBERO-PRO (0.59), and Polaris (0.607), with full results in Appendix A.9. On SimplerEnv, interleaved text-action supervision improves the overall average from 74.9% to 76.7%; further co-training comparisons appear in Appendix A.3.

Real-world evaluation. SemAE reaches 73.7% success on Franka and 89.4% on WidowX 250s, compared with 65.9% and 84.1% for $\pi_{0.5}$ (Tables 1 and 2). On DROID, the average increases from 0.77 to 0.81 (Table 22). These results establish strong manipulation performance across settings; the following analysis examines how performance varies with language requirements.

Table 2: **Real-world Franka results.** We report task success rates (%) across six task groups, with intermediate grasp success for Kitchen and Three Cube. The overall averages summarize policy performance across object manipulation, drawer closing, charger insertion, and tea preparation.

Model	Kitchen		Three Cube		Plush Toy	Close Drawer	Insert Charger	Make Tea	#Overall
	Grasp Object	Success	Grasp Cube	Success	Success	Success	Success	Success	Average
Diffusion Policy (Chi et al., 2023)	77.3%	72.7%	6.1%	3.0%	45.5%	45.5%	18.2%	0.0%	30.8%
OCTO (Octo Model Team et al., 2024)	36.4%	31.8%	36.4%	36.4%	27.3%	27.3%	18.2%	0.0%	23.5%
OpenVLA (Kim et al., 2024)	72.7%	54.5%	51.5%	39.4%	45.5%	54.5%	9.1%	9.1%	35.4%
SpatialVLA (Qu et al., 2025b)	86.4%	86.4%	60.6%	54.5%	63.6%	36.4%	0.0%	9.1%	41.7%
π_0 (Black et al., 2024)	90.9%	90.9%	60.6%	60.6%	72.7%	63.6%	36.4%	18.2%	57.1%
π_0-FAST (Pertsch et al., 2025)	40.9%	31.8%	33.3%	30.3%	45.5%	36.4%	0.0%	0.0%	24.0%
GR00T (Bjorck et al., 2025)	72.7%	68.2%	48.5%	48.5%	54.5%	63.6%	18.2%	0.0%	42.2%
OpenHelix (Cui et al., 2025)	45.5%	40.9%	54.5%	51.5%	63.6%	54.5%	9.1%	0.0%	36.6%
$\pi_{0.5}$ (Intelligence et al., 2025)	**100.0%**	95.5%	69.7%	63.6%	81.8%	81.8%	36.4%	**36.4%**	65.9%
SemAE	**100.0%**	**100.0%**	**78.8%**	**78.8%**	**90.9%**	**90.9%**	**45.5%**	**36.4%**	**73.7%**

4.2 LANGUAGE-FOLLOWING ABILITY

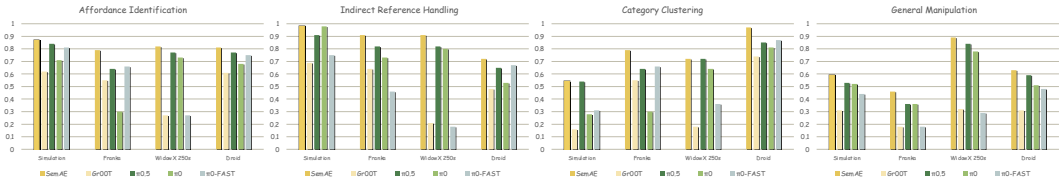


Figure 5: **Language following across four evaluation settings.** The first three panels report success rates for Affordance Identification, Indirect Reference Handling, and Category Clustering in Simulation, Franka, WidowX 250s, and DROID. For each model and metric, the four setting-specific rates are averaged with equal weight to obtain the score in Figure 1(d). The fourth panel reports general manipulation performance separately.

Aggregate performance. Averaging the four platform-specific rates in Figure 5 gives SemAE scores of 82.25%, 88.00%, and 75.50% for Affordance Identification, Indirect Reference Handling, and Category Clustering. The corresponding $\pi_{0.5}$ scores are 75.50%, 80.00%, and 68.75%, yielding improvements of **6.75, 8.00, and 6.75 percentage points**. Equal platform weights ensure that these scores summarize the four settings without letting one platform dominate through a larger number of trials.

Platform-specific results. SemAE improves Affordance Identification over $\pi_{0.5}$ in all four settings, with gains ranging from 3 percentage points in Simulation to 15 on Franka. Indirect Reference Handling improves by 7–9 points in every setting, showing that its 8-point aggregate gain is shared across all four platforms. Category Clustering gains are concentrated on Franka and DROID, at 15 and 12 points, respectively; SemAE matches $\pi_{0.5}$ in Simulation and on WidowX 250s. The aggregate category score therefore reflects improvements on these two real-world settings, whereas the other two metrics improve across all four.

The fourth panel reports higher general manipulation success for SemAE than for $\pi_{0.5}$ in every setting. Together, the panels show improved language following alongside strong manipulation performance. They also distinguish consistent gains in object selection and indirect references from the more platform-dependent gains in category-level grounding. Task groups for the three metrics are listed in Appendix A.1.

Attention analysis. Figure 6 compares the attention scores of $\pi_{0.5}$ and SemAE on text and action tokens under identical inputs. SemAE assigns higher attention scores to text tokens in the displayed examples. This visualization complements the behavioral evaluation but does not by itself establish semantic preservation.

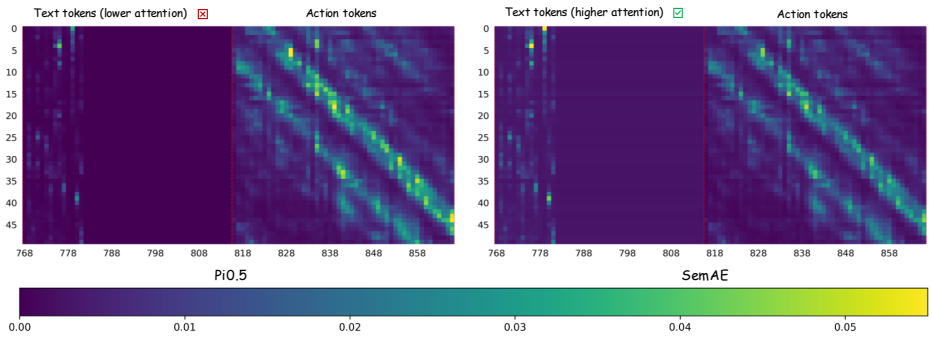


Figure 6: **Attention maps of SemAE and $\pi_{0.5}$.** Given identical inputs, we compare the attention scores on text and action tokens. SemAE assigns higher attention scores to text tokens than $\pi_{0.5}$.

4.3 Ablation Study

We remove one component at a time on Polaris, with per-task results in Table 21. The full model achieves 0.607 average success; removing distillation, flow decoding, or parallel decoding reduces it to 0.487, 0.540, or 0.562, respectively.

Layer-wise motion primitive distillation. Training SemAE with reconstruction alone causes the largest average drop, at 12.0 percentage points, while retaining the flow decoder. The loss is especially large on BlockStackKitchen (0.57 to 0.37) and FoodBussing (0.54 to 0.38), but small on PanClean (0.57 to 0.55) and TapeIntoContainer (0.85 to 0.84). Thus, the benefit is task-dependent, and the unchanged decoder architecture shows that a flow decoder alone does not account for the full model's gains. This supports the contribution of the distillation objective, although task success alone cannot separate improved instruction grounding from improved execution.

Patch-wise action flow decoding. Replacing the flow decoder with an MLP reduces average success by 6.7 points, with lower scores on all six tasks. This complements the distillation ablation: learning semantic-aligned action embeddings and decoding them into continuous actions both contribute to performance. The pattern is consistent with a generative decoder being better suited to representing detailed action distributions; additional decoder comparisons appear in Appendix A.6.

Semantic parallel decoding. Autoregressive prediction of semantic-aligned action embeddings lowers average success by 4.5 points. BlockStackKitchen and MoveLatteCup remain unchanged, while the other four tasks decline, showing that the benefit is not uniform across tasks. Parallel prediction therefore improves overall success in addition to generating all action-slot embeddings in one forward pass. Further decoding-strategy and latency comparisons are provided in Appendix A.4.

5 Conclusion

We presented **SemAE**, an encoder–decoder action expert that aligns continuous robot actions with the semantic space of a pretrained VLM via *layer-wise motion primitive distillation*, together with a *patch-wise rectified-flow* decoder that efficiently generates high-dimensional action chunks. Built on SemAE, **SemVLA** uses *semantic parallel decoding* with dedicated action slots to predict semantic action embeddings in one forward pass and decode them into continuous actions, improving language following without sacrificing manipulation performance. Across standard simulation benchmarks and real-robot evaluations, SemVLA achieves strong overall success and markedly better language grounding under ambiguity. One limitation is that Stage 1 training projects robot trajectories onto observation images and therefore requires additional information about the robot camera's intrinsic and extrinsic parameters.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

Suneel Belkhale and Dorsa Sadigh. Minivla: A better vla with a smaller footprint, 2024. URL https://github.com/Stanford-ILIAD/openvla-mini.

Johan Bjorck, Fernando Castañeda, Nikita Cherniadev, Xingye Da, Runyu Ding, Linxi Fan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, et al. Gr00t n1: An open foundation model for generalist humanoid robots. *arXiv preprint arXiv:2503.14734*, 2025.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Qingwen Bu, Jisong Cai, Li Chen, Xiuqi Cui, Yan Ding, Siyuan Feng, Shenyuan Gao, Xindong He, Xu Huang, Shu Jiang, et al. Agibot world colosseo: A large-scale manipulation platform for scalable and intelligent embodied systems. *arXiv preprint arXiv:2503.06669*, 2025a.

Qingwen Bu, Yanting Yang, Jisong Cai, Shenyuan Gao, Guanghui Ren, Maoqing Yao, Ping Luo, and Hongyang Li. Univla: Learning to act anywhere with task-centric latent actions. *arXiv preprint arXiv:2505.06111*, 2025b.

Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, et al. Pali-x: On scaling up a multilingual vision and language model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

Open X-Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, et al. Open x-embodiment: Robotic learning datasets and rt-x models. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

Can Cui, Pengxiang Ding, Wenxuan Song, Shuanghao Bai, Xinyang Tong, Zirui Ge, Runze Suo, Wanqi Zhou, Yang Liu, Bofang Jia, et al. Openhelix: A short survey, empirical analysis, and open-source dual-system vla model for robotic manipulation. *arXiv preprint arXiv:2505.03912*, 2025.

Danny Driess, Jost Tobias Springenberg, Brian Ichter, Lili Yu, Adrian Li-Bell, Karl Pertsch, Allen Z. Ren, Homer Walke, Quan Vuong, Lucy Xiaoyang Shi, and Sergey Levine. Knowledge insulating vision-language-action models: Train fast, run fast, generalize better. *arXiv preprint*, 2025.

Jiayuan Gu, Sean Kirmani, Paul Wohlhart, Yao Lu, Montserrat Gonzalez Arenas, Kanishka Rao, Wenhao Yu, Chuyuan Fu, Keerthana Gopalakrishnan, Zhuo Xu, Priya Sundaresan, Peng Xu, Hao Su, Karol Hausman, Chelsea Finn, Quan Vuong, and Ted Xiao. Robotic task generalization via hindsight trajectory sketches. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=F1TKzG8LJO. arXiv:2311.01977.

Chia-Yu Hung, Navonil Majumder, Haoyuan Deng, Liu Renhang, Yankang Ang, Amir Zadeh, Chuan Li, Dorien Herremans, Ziwei Wang, and Soujanya Poria. Nora-1.5: A vision-language-action model trained using world model- and action-based preference rewards, 2025a. URL https://arxiv.org/abs/2511.14659.

Chia-Yu Hung, Qi Sun, Pengfei Hong, Amir Zadeh, Chuan Li, U-Xuan Tan, Navonil Majumder, and Soujanya Poria. Nora: A small open-sourced generalist vision language action model for embodied tasks, 2025b. URL https://arxiv.org/abs/2504.19854.

Physical Intelligence, Kevin Black, Noah Brown, James Darpinian, Karan Dhabalia, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Manuel Y. Galliker, Dibya Ghosh, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Devin LeBlanc, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Allen Z. Ren, Lucy Xiaoyang Shi, Laura Smith, Jost Tobias Springenberg, Kyle Stachowicz, James Tanner, Quan Vuong, Homer Walke, Anna Walling, Haohuan Wang, Lili Yu, and Ury Zhilinsky. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Arhan Jain, Mingtong Zhang, Kanav Arora, William Chen, Marcel Torne, Muhammad Zubair Irshad, Sergey Zakharov, Yue Wang, Sergey Levine, Chelsea Finn, et al. Polaris: Scalable real-to-sim evaluations for generalist robot policies. *arXiv preprint arXiv:2512.16881*, 2025.

Siddharth Karamcheti, Suraj Nair, Ashwin Balakrishna, Percy Liang, Thomas Kollar, and Dorsa Sadigh. Prismatic vlms: Investigating the design space of visually-conditioned language models. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2024.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Moo Jin Kim, Chelsea Finn, and Percy Liang. Fine-tuning vision-language-action models: Optimizing speed and success. *arXiv preprint arXiv:2502.19645*, 2025.

Qixiu Li, Yaobo Liang, Zeyu Wang, Lin Luo, Xi Chen, Mozheng Liao, Fangyun Wei, Yu Deng, Sicheng Xu, Yizhong Zhang, et al. Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation. *arXiv preprint arXiv:2411.19650*, 2024a.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. In *Advances in Neural Information Processing Systems*, 2024b.

Xinghang Li, Peiyan Li, Minghuan Liu, Dong Wang, Jirong Liu, Bingyi Kang, Xiao Ma, Tao Kong, Hanbo Zhang, and Huaping Liu. Towards generalist robot policies: What matters in building vision-language-action models. *arXiv preprint arXiv:2412.14058*, 2024c.

Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2024d.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *arXiv preprint arXiv:2306.03310*, 2023.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022.

Yicheng Liu, Shiduo Zhang, Zibin Dong, Baijun Ye, Tianyuan Yuan, Xiaopeng Yu, Linqi Yin, Chenhao Lu, Junhao Shi, Luca Jiang-Tao Yu, Liangtao Zheng, Tao Jiang, Jingjing Gong, Xipeng Qiu, and Hang Zhao. Faster: Toward efficient autoregressive vision language action modeling via neural action tokenization, 2025. URL https://arxiv.org/abs/2512.04952.

Huaihai Lyu, Chaofan Chen, Senwei Xie, Pengwei Wang, Xiansheng Chen, Shanghang Zhang, and Changsheng Xu. Omnisat: Compact action token, faster auto regression, 2025. URL https://arxiv.org/abs/2510.09667.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

Karl Pertsch, Kyle Stachowicz, Brian Ichter, Danny Driess, Suraj Nair, Quan Vuong, Oier Mees, Chelsea Finn, and Sergey Levine. Fast: Efficient action tokenization for vision-language-action models. *arXiv preprint arXiv:2501.09747*, 2025.

Delin Qu, Haoming Song, Qizhi Chen, Zhaoqing Chen, Xianqiang Gao, Modi Shi, Guanghui Ren, Maoqing Yao, Bin Zhao, and Dong Wang. Eo-1: Interleaved vision-text-action pretraining for general robot control, 2025a. Technical report.

Delin Qu, Haoming Song, Qizhi Chen, Yuanqi Yao, Xinyi Ye, Yan Ding, Zhigang Wang, JiaYuan Gu, Bin Zhao, Dong Wang, et al. Spatialvla: Exploring spatial representations for visual-language-action model. *arXiv preprint arXiv:2501.15830*, 2025b.

Gemini Robotics Team, Saminda Abeyruwan, Joshua Ainslie, Jean-Baptiste Alayrac, Montserrat Gonzalez Arenas, Travis Armstrong, Ashwin Balakrishna, Robert Baruch, Maria Bauza, Michiel Blokzijl, et al. Gemini robotics: Bringing ai into the physical world. *arXiv preprint arXiv:2503.20020*, 2025.

Homer Walke, Kevin Black, Abraham Lee, Moo Jin Kim, Max Du, Chongyi Zheng, Tony Zhao, Philippe Hansen-Estruch, Quan Vuong, Andre He, Vivek Myers, Kuan Fang, Chelsea Finn, and Sergey Levine. Bridgedata v2: A dataset for robot learning at scale. In *Proceedings of the Conference on Robot Learning (CoRL)*, 2023.

Lirui Wang, Xinlei Chen, Jialiang Zhao, and Kaiming He. Scaling proprioceptive-visual learning with heterogeneous pre-trained transformers. *Advances in Neural Information Processing Systems*, 2024.

Yating Wang, Haoyi Zhu, Mingyu Liu, Jiange Yang, Hao-Shu Fang, and Tong He. Vq-vla: Improving vision-language-action models via scaling vector-quantized action tokenizers. *arXiv preprint arXiv:2507.01016*, 2025.

Junjie Wen, Yichen Zhu, Jinming Li, Zhibin Tang, Chaomin Shen, and Feifei Feng. Dexvla: Vision-language model with plug-in diffusion expert for general robot control. *arXiv preprint arXiv:2502.05855*, 2025.

Kun Wu, Chengkai Hou, Jiaming Liu, Zhengping Che, Xiaozhu Ju, Zhuqin Yang, Meng Li, Yinuo Zhao, Zhiyuan Xu, Guang Yang, et al. Robomind: Benchmark on multi-embodiment intelligence normative data for robot manipulation. *arXiv preprint arXiv:2412.13877*, 2024.

Zhengrong Yue, Haiyu Zhang, Xiangyu Zeng, Boyu Chen, Chenting Wang, Shaobin Zhuang, Lu Dong, KunPeng Du, Yi Wang, Limin Wang, and Yali Wang. Uniflow: A unified pixel flow tokenizer for visual understanding and generation. *arXiv preprint*, 2026.

Ruijie Zheng, Yongyuan Liang, Shuaiyi Huang, Jianfeng Gao, Hal Daumé III, Andrey Kolobov, Furong Huang, and Jianwei Yang. Tracevla: Visual trace prompting enhances spatial-temporal awareness for generalist robotic policies. *arXiv preprint arXiv:2412.10345*, 2024.

Hongyi Zhou, Weiran Liao, Xi Huang, Yucheng Tang, Fabian Otto, Xiaogang Jia, Xinkai Jiang, Simon Hilber, Ge Li, Qian Wang, Ömer Erdinç Yağmurlu, Nils Blank, Moritz Reuss, and Rudolf Lioutikov. Beast: Efficient tokenization of b-splines encoded action sequences for imitation learning, 2025. URL https://arxiv.org/abs/2506.06072.

A SUPPLEMENTARY MATERIAL

This supplementary material complements the main paper by providing a deeper investigation into the design principles and capability boundaries of SemAE. We structure the appendix as follows: Appendix A.1 presents task groups for the language-following metrics defined in Section 4; the results and attention analysis are presented in Section 4.2. Appendix A.2 presents a multi-dimensional comparison between SemAE and the baseline $\pi_{0.5}$-KI to provide further insights into the language following challenges in Vision-Language-Action models. Appendix A.3 explores the scalability of SemAE to text-action co-training, demonstrating the potential of aligning robot actions with the semantic space for large-scale text-action pretraining of manipulation foundation models. We then provide detailed ablation studies on key architectural components: Appendix A.4 analyzes four paradigms of interaction between SemAE and pretrained VLMs to balance performance and efficiency; Appendix A.5 evaluates three motion primitive distillation strategies, highlighting the efficacy of Layer-Wise Adaptive Distillation in encoding and aligning motion primitives; Appendix A.6 validates the Patch-wise Action Flow Decoder's ability to generate continuous robot actions from semantic embeddings; and Appendix A.7 examines three robot state representation paradigms, demonstrating their impact on language following performance. Finally, Appendix A.8 provides implementation details, hyperparameters, and computational costs; Appendix A.9 collects the full simulation benchmark results; and Appendix A.10 describes the benchmarks, baseline policies, and real-world evaluation protocols.

A.1 LANGUAGE-FOLLOWING TASK MAPPING

The three language-following metrics are defined in Section 4. Table 3 organizes tasks and task groups by metric across Simulation, Franka, WidowX 250s, and DROID. Detailed benchmark and real-world task descriptions are provided in Appendices A.10.1 and A.10.3.

Table 3: **Language-following task mapping.** Tasks and task groups organized by metric across four evaluation settings.

Setting	Metric	Tasks or task groups
Simulation	Affordance Identification	LIBERO-Object suite (Table 18)
Simulation	Indirect Reference Handling	LIBERO-PRO: Sem shifts in all four suites (Table 19)
Simulation	Category Clustering	Polaris: Food Bussing; Organize Tools (Table 21)
Franka	Affordance Identification	Three Cube (Table 2)
Franka	Indirect Reference Handling	Plush Toy (Table 2)
Franka	Category Clustering	Kitchen (Table 2)
WidowX 250s	Affordance Identification	Put Carrot on Plate (Table 1)
WidowX 250s	Indirect Reference Handling	Put Green Cup on Pink Cloth (Table 1)
WidowX 250s	Category Clustering	Put Purple Cup on White Plate (Table 1)
DROID	Affordance Identification	Flip Bottle; Add Ingredients; Cover the Pot; Put Spoon; Put Bowl; Pick Bread; Arrange Cube (Table 22)
DROID	Indirect Reference Handling	Arrange Cube (Table 22)
DROID	Category Clustering	Pick Bread (Table 22)

A.2 COMPARISONS BETWEEN SEMAE AND $\pi_{0.5}$-KI

$\pi_{0.5}$-KI serves as a formidable baseline in the domain of Vision-Language-Action models, employing stop-gradient mechanisms and co-training strategies to significantly bolster language following capabilities. Its single-stage training paradigm is computationally efficient, positioning it as a strong candidate for scalable robot learning. While SemAE demonstrates superior language following performance, it adopts a two-stage training framework that involves first training a semantic-aligned action expert followed by VLM backbone integration. This results in a computational demand approximately $1.8\times$ to $2.3\times$ higher than that of $\pi_{0.5}$-KI.

Despite its efficiency, $\pi_{0.5}$-KI exhibits a high dependency on task-relevant textual data during the training phase. We observe a distinct performance gap in the language following abilities of $\pi_{0.5}$-KI between the DROID and WidowX evaluation scenarios. In the DROID setting, the model benefits from the official Physical Intelligence checkpoint, which is trained using a mixture of text and robot data, resulting in strong instruction adherence. In contrast, for the WidowX scenario, we fine-tuned the model on the Bridge dataset using the official open-source repository. Due to the lack of implementation for text data co-training in the official open source code, the model on the WidowX was trained exclusively on robot action data.

This reliance on pure robot action data during fine-tuning leads to a degradation of the semantic knowledge embedded within the VLM backbone of $\pi_{0.5}$-KI. Without the regularizing effect of textual data, the model suffers from catastrophic forgetting of linguistic concepts. Conversely, SemAE mitigates this issue by explicitly aligning robot actions with the VLM's semantic space during the first stage of training. This alignment ensures that even when the model is post-trained on downstream datasets containing only robot actions, the semantic integrity of the VLM is preserved. Consequently, SemAE maintains robust language following capabilities and minimizes the erosion of semantic knowledge, demonstrating greater stability in data-constrained fine-tuning regimes.

A.3 Scaling to Text-Action Co-training

Recent studies (Intelligence et al., 2025; Driess et al., 2025; Team et al., 2025) demonstrate that co-training with a mixture of text and action data improves robotic control performance while preserving the language capabilities of the VLM backbone. Furthermore, incorporating task-relevant visual grounding data, such as trajectory point prediction, bounding box annotation, and multi-view keypoint matching, enables the VLM to generate latent representations better suited for manipulation tasks. Since SemAE effectively aligns high-dimensional continuous robot actions with the semantic space of the VLM, we further explore its scalability under this text-action co-training paradigm. We select a subset of the EO-DATA-1.5M dataset (Qu et al., 2025a) for training. EO-DATA-1.5M provides large-scale annotations for manipulation tasks across datasets such as Agibot-World (Bu et al., 2025a), OpenX-Embodiment (Collaboration et al., 2024), and Robomind (Wu et al., 2024). It covers diverse tasks including Visual Grounding, Object Referring, Object Pointing, Multi-view Pointing, and Task Planning. Specifically, we utilize the interleaved manipulation data constructed from Bridge Data V2 (Walke et al., 2023) and Fractal (Brohan et al., 2022), as illustrated in Figure 7.

To quantify the benefits of co-training, we evaluate the models on the SimplerEnv benchmark. As shown in Table 4, we compare the performance of SemAE trained solely on robot action data, SemAE w/ co-training trained with additional interleaved text data, and the EO-1 baseline (Qu et al., 2025a) pre-trained on the comprehensive EO-DATA-1.5M dataset. The results indicate that SemAE w/ co-training achieves a consistent performance gain over the standard SemAE. This improvement verifies the scalability of our approach, demonstrating that incorporating text data enriches the semantic context and enhances manipulation capabilities. Furthermore, SemAE w/ co-training surpasses EO-1. Since both models utilize interleaved text-action co-training, the superior performance of SemAE highlights the critical role of aligning robot actions with the VLM's semantic space. By mapping actions into the semantic space, SemAE enables the model to more effectively process the semantic information within robot actions, leading to improved performance on manipulation tasks.

A.4 Analysis of Semantic Parallel Decoding

The interaction paradigm between the action expert and the VLM backbone is pivotal for developing Vision-Language-Action (VLA) models with robust language following capabilities. An optimal design must seamlessly transfer action-relevant representations from the VLM backbone to the action expert while simultaneously optimizing inference efficiency. Prior works, such as π_0 (Intelligence et al., 2025), achieve a balance between performance and efficiency by introducing cross-attention mechanisms. By caching the Key-Value (KV) states of the VLM backbone, this approach efficiently transfers action representations; since the action expert requires multiple iterative forward passes during flow matching, reusing these cached states circumvents the need to repeatedly execute the parameter-heavy VLM backbone, thereby ensuring inference speed. However, while efficient, this tightly coupled design introduces a significant drawback: gradients back-propagated from the ac-

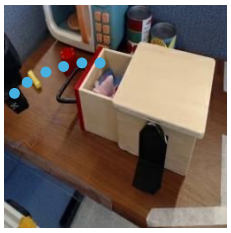
Grounding QA



User: *<image>For the robotic arm to perform \"reach into the drawer to grab the pink and white toy\", how should it be maneuvered? Specify trajectory points in XML format.*

GPT: *```xml\n<points x1=\"164\" y1=\"187\" x2=\"152\" y2=\"170\" x3=\"142\" y3=\"154\" x4=\"132\" y4=\"139\" x5=\"123\" y5=\"125\" x6=\"114\" y6=\"104\"></points>\n```*

Robot Control



User: *<image><state>How should the robotic manipulator execute the trajectory ```xml\n<points x1=\"164\" y1=\"187\" x2=\"152\" y2=\"170\" x3=\"142\" y3=\"154\" x4=\"132\" y4=\"139\" x5=\"123\" y5=\"125\" x6=\"114\" y6=\"104\"></points>\n``` physically?*

GPT: *<VLA><action>*

Task QA



User: *<image>Has the robot finished \"reach into the drawer to grab the pink and white toy\"?"*

GPT: No.

Figure 7: **Interleaved text-action supervision.** An example from EO-DATA-1.5M (Qu et al., 2025a) interleaves visual grounding QA, robot control, and task-completion QA. Text gives trajectory points and completion status; actions are conditioned on the image, state, and trajectory.

Table 4: **Text-action co-training.** Success rates (%) on SimplerEnv for EO-1 and SemAE with or without interleaved text-action co-training. Results cover WidowX and two Google Robot protocols: Visual Matching and Variant Aggregation. Each block reports per-task results and their average.

WidowX Benchmark	**Spoon**	**Carrot**	**Stack**	**Eggplant**	**Average**
EO-1 (Qu et al., 2025a)	63.6%	54.5%	**81.8%**	90.9%	72.7%
SemAE	88.7%	89.2%	79.8%	94.3%	88.0%
SemAE w/ co-training	**89.3%**	**91.3%**	78.3%	**95.8%**	**88.7%**
Google EDR (Matching)	**Coke Can**	**Move Near**	**Drawer**	**Apple**	**Average**
EO-1 (Qu et al., 2025a)	98.0%	**83.8%**	71.3%	52.8%	76.5%
SemAE	**98.2%**	81.2%	67.9%	49.3%	74.2%
SemAE w/ co-training	**98.2%**	83.7%	**72.5%**	**54.1%**	**77.1%**
Google EDR (Aggregation)	**Coke Can**	**Move Near**	**Drawer**	**Apple**	**Average**
EO-1 (Qu et al., 2025a)	91.6%	81.7%	55.0%	**23.8%**	63.0%
SemAE	91.7%	82.1%	54.3%	22.4%	62.6%
SemAE w/ co-training	**93.2%**	**84.6%**	**56.2%**	23.6%	**64.4%**

tion expert during training can disrupt the pre-trained VLM weights. This interference degrades the VLM's inherent semantic knowledge, ultimately impairing the model's language following performance. Consequently, exploring interaction paradigms that maintain efficiency without compromising semantic integrity is of paramount importance. To comprehensively evaluate different interaction paradigms, we investigate four decoding strategies and report their language following capabilities and inference latencies in Table 5.

Table 5: **Semantic action decoding.** We compare four ways to predict latent action embeddings. Language Following averages Affordance Identification, Indirect Reference Handling, and Category Clustering. Latency measures VLM runtime for embedding prediction, excluding action decoding.

Interaction Paradigm	**Affordance**	**Reference**	**Category**	**Language Following**	**Latency**
Parallel Decoding	0.76	**0.65**	0.83	0.75	**19.1ms**
Latent Flow Matching	**0.78**	0.64	**0.89**	**0.77**	178.6ms
Autoregressive	0.54	0.47	0.71	0.57	63.7ms
Masked Autoregressive	0.71	0.58	0.77	0.69	83.4ms

Parallel Decoding. This is the default decoding strategy used in SemVLA. As illustrated in Figure 3, it utilizes a set of learnable query tokens to extract semantic action embeddings. During inference, the VLM prefills these tokens and predicts all semantic action embeddings in a single forward pass. This approach, which is detailed in Section 3.2, ensures efficient semantic information transfer without iterative overhead.

Latent Flow Matching. This paradigm replaces the learnable query tokens with Gaussian noise sampled from a normal distribution. The entire VLM backbone functions as a flow velocity field predictor. During decoding, the model performs multiple iterative forward passes to denoise the initial noise into valid latent action embeddings. This essentially models the generation of semantic action embeddings as a flow matching process conditioned on the visual-textual context.

Autoregressive. This baseline employs the traditional VLM decoding strategy. It necessitates discretizing the continuous latent action embeddings into a finite vocabulary. The VLM then autoregressively predicts the discrete tokens one by one. This method strictly adheres to the standard language modeling objective but introduces inevitable quantization errors.

Masked Autoregressive (MAR). Inspired by recent advancements in continuous image generation (Li et al., 2024b), MAR provides a framework for autoregressively modeling continuous data without vector quantization. We introduce a lightweight MLP head on top of the VLM backbone. Unlike standard autoregressive models that output categorical distributions over a discrete vocabulary, MAR models the conditional probability of the next continuous embedding via a denoising

diffusion process. Specifically, at each step, the VLM predicts a conditioning vector, and the MLP head executes a reverse diffusion process to sample the continuous semantic embedding. This allows for the generation of continuous embeddings while maintaining the sequential dependency of autoregression.

As evidenced in Table 5, Latent Flow Matching achieves strong language following performance (0.77) by leveraging iterative refinement to capture precise semantic information. However, its requirement for multiple full-backbone forward passes results in a prohibitive latency of 178.6ms. Conversely, the Autoregressive baseline exhibits a significant performance drop (0.57), primarily due to the information loss inherent in discretizing continuous action semantics. Masked Autoregressive mitigates this by modeling continuous embeddings, improving language following to 0.69. Nevertheless, its sequential prediction nature still incurs a notable latency cost (83.4ms). Parallel Decoding emerges as the optimal solution, delivering a language following capability (0.75) comparable to Latent Flow Matching while achieving the lowest latency (19.1ms). This balance confirms that Semantic Parallel Decoding effectively aligns robot actions with VLM semantics without compromising real-time responsiveness.

A.5 Analysis of Motion Primitive Distillation

The design of Motion Primitive Distillation is pivotal for the training of SemAE. Traditional VAE-based action tokenizers, such as VQ-VLA (Wang et al., 2025), primarily focus on compressing robot actions into compact embeddings that retain sufficient information for reconstruction. In contrast, SemAE must not only ensure reconstructability but also extract semantic-rich information from robot actions. This semantic extraction allows the encoder to generate latent action embeddings that align with the VLM's semantic space, enabling the VLM to process these embeddings as seamlessly as natural language tokens and effectively utilize its pre-trained semantic knowledge. To validate our design, we investigate the impact of three distillation strategies on language following capabilities and dexterous manipulation capabilities, as detailed in Table 6:

Table 6: **Motion primitive distillation.** We compare adaptive layer weights, constant layer weights, and final-layer-only distillation. Affordance Identification, Indirect Reference Handling, Category Clustering, and Dexterous Manipulation scores assess language following and motor control.

Distillation Paradigm	Affordance	Reference	Category	Dexterous Manipulation
Adaptive Weight	**0.76**	**0.65**	**0.83**	**0.91**
Constant Weight	0.75	0.64	**0.83**	0.84
Final-Only	0.69	0.61	0.77	0.89

Adaptive Weight. This is the default strategy employed in SemAE. It dynamically adjusts distillation weights based on the depth of the Transformer layers. Inspired by the Layer-Wise Adaptive Self-Distillation in UniFlow (Yue et al., 2026), we recognize that different Transformer layers capture distinct features: shallow layers excel at extracting fine-grained motion primitives, while deeper layers are more adept at expressing high-level semantic information. Consequently, as described in Section 3.1, we assign higher distillation weights to deeper layers. This encourages the encoder's shallow layers to focus on capturing motion primitives from robot actions, while guiding the deeper layers to align these primitives with the semantic representations of the teacher VLM backbone.

Constant Weight. This strategy applies a uniform distillation weight across all Transformer layers. The objective is to enforce a strict alignment between every hidden state of the SemAE encoder and the corresponding layer of VLM teacher.

Final-Only. This approach distills only the final Transformer layer, a paradigm commonly used in image understanding tasks to align the output representation with a target space.

The experimental results in Table 6 demonstrate that the Adaptive Weight strategy yields the best performance in both language following and dexterous manipulation tasks. Conversely, the Constant Weight strategy leads to a slight decline in language following and a significant drop in dexterous manipulation. This suggests that enforcing strict alignment at every layer over-regularizes the model, disrupting the encoder's ability to preserve low-level motion primitives necessary for precise control. Furthermore, the Final-Only strategy results in a minor decrease in manipulation performance but

a substantial degradation in language following. This indicates that the supervision signal from the final layer alone is insufficient to stabilize the optimization of the encoder, failing to produce consistent latent action embeddings aligned with teacher VLM.

A.6 Analysis of Patch-wise Action Flow Decoding

The decoder is the component that is always executed at test time. It must translate a compact set of semantic action embeddings, predicted in the VLM semantic space, into high-dimensional continuous robot actions. This translation is non-trivial because the embedding has to preserve two properties at once. It must retain fine-grained motion details for accurate control, and it must keep instruction-relevant semantics so that the generated trajectory matches the language intent. Recent VLAs also suggest that flow matching is a strong choice for modeling continuous robot actions at the chunk level. We therefore implement the SemAE decoder as a lightweight rectified-flow model and analyze how different decoding granularities affect capability and efficiency. Table 7 compares three decoding paradigms, evaluated by dexterous manipulation, language following, and decoder latency. Dexterous manipulation measures control accuracy, language following measures whether decoded trajectories reflect the instruction-relevant semantics in the embeddings, and latency reports the decoder runtime that matters for closed-loop robot control.

Table 7: **Action decoder design.** We compare patch-wise flow decoding, step-wise flow decoding, and a parameter-matched MLP using Dexterous Manipulation and Language Following scores. Latency measures decoder runtime for generating actions conditioned on latent action embeddings.

Decoding Paradigm	Dexterous Manipulation	Language Following	Latency
Patch-wise Decoding	**0.91**	**0.75**	35.9ms
Step-wise Decoding	0.82	0.65	41.3ms
MLP	0.42	0.21	**7.3ms**

Patch-wise Decoding. This is our default design. We partition each action chunk into S non-overlapping action patches and decode each patch $\mathbf{a}^{(i)} \in \mathbb{R}^{p \times d_a}$ conditioned on one embedding $\mathbf{c}^{(i)}$. The patch length p follows the encoder compression ratio. In our setup, the action-chunk downsampling rate is 8, so each latent embedding can decode an 8-step action patch.

Step-wise Decoding. This baseline adds a 1D transposed-convolution upsampling module that expands the S patch-level embeddings into T step-level embeddings. The decoder then uses these step-level embeddings as rectified-flow conditions during iterative denoising, so every action step is paired with a distinct latent condition. This increases the effective sequence length inside the action decoder and adds extra computation.

MLP. This baseline replaces the flow decoder with a 3-layer MLP that directly maps the semantic action embeddings to the full action chunk. For a fair comparison, we match the parameter count of MLP decoder to the flow-based decoder.

The results in Table 7 show that Patch-wise Decoding achieves the best overall trade-off, with 0.91 dexterous manipulation and 0.75 language following at 35.9ms latency. MLP is faster at 7.3ms, but its performance drops sharply to 0.42 and 0.21, suggesting that a direct deterministic mapping is not expressive enough to decode semantic embeddings into accurate continuous action chunks. Step-wise Decoding is slower than patch-wise, with latency increasing from 35.9ms to 41.3ms, and it also degrades both dexterous manipulation and language following, from 0.91 to 0.82 and from 0.75 to 0.65. Compared with Step-wise Decoding, Patch-wise Decoding improves dexterous manipulation by 0.09 and language following by 0.10, while reducing latency by 5.4ms. Compared with MLP, Patch-wise Decoding improves dexterous manipulation by 0.49 and language following by 0.54, at the cost of 28.6ms higher latency. These results support modeling actions at a short-horizon patch granularity. They are also consistent with the observation in Diffusion Policy (Chi et al., 2023) that denoising-based action models benefit from predicting multi-step action chunks, since chunk-level generation better captures temporal correlations than per-step modeling under same compute cost.

Table 8: **Robot state representation.** We compare continuous state embeddings, discretized states serialized as text, and discretized states represented by special tokens. Affordance Identification, Indirect Reference Handling, Category Clustering, and Dexterous Manipulation scores are reported.

State Representation	Affordance	Reference	Category	Dexterous Manipulation
Continuous State	0.76	**0.65**	**0.83**	**0.91**
Text State	**0.79**	0.63	0.78	0.87
Special Token State	0.71	0.59	0.81	0.83

A.7 Analysis of State Representation

Robot state representation also shapes the capability profile of a Vision-Language-Action model. In many manipulation settings, robot states are continuous proprioceptive signals that are closely coupled with actions. An appropriate representation can therefore help the VLM backbone better interpret the current physical configuration and generate actions that both follow language and satisfy the required control precision. Prior work such as KI (Driess et al., 2025) reports that representing robot states as discrete textual tokens can mitigate language-following degradation observed in continuous VLAs such as π_0 (Black et al., 2024). To study this design choice in SemAE, we compare three state representation paradigms in Table 8, evaluated on language following and dexterous manipulation capabilities. **Continuous State.** We encode the raw continuous robot state into one continuous embedding and append it to the input sequence. This representation preserves fine-grained proprioceptive information, which is crucial for precise control.

Text State. Following KI (Driess et al., 2025), we discretize each scalar state dimension into 256 bins and serialize the discretized values into a natural-language-style string. The state is then appended as regular text tokens. This makes the state more consistent with the text input format of VLMs, but it introduces quantization error and increases the number of input tokens.

Special Token State. We also discretize each state dimension into bins, but represent each discretized value using dedicated special tokens with learned embeddings. Compared with Text State, this avoids subword tokenization artifacts and yields a fixed, structured layout, while still incurring discretization.

Table 8 shows that Continuous State performs best overall in SemAE. It achieves the highest Indirect Reference Handling and Category Clustering scores, and it yields the strongest dexterous manipulation performance. Text State slightly improves Affordance Identification, but it degrades dexterous manipulation, likely due to the loss of fine-grained numeric precision after discretization. Special Token State underperforms in both language following and manipulation, suggesting that discretization alone is insufficient without a representation that is well aligned with the VLM backbone. This trend differs from the conclusion in KI (Driess et al., 2025), which highlights the benefit of text-based states for preserving language capability during adaptation. We attribute the difference to the interface used in SemAE. SemAE encodes continuous robot actions into continuous semantic action embeddings that are supervised to align with VLM hidden states. Under this design, representing robot states as continuous embeddings allows state information to interact with the same semantic space as actions and language, enabling more direct transfer of instruction-relevant semantics between state and action generation. In contrast, discretized state tokens introduce a representational gap that can hinder this semantic transfer, even if they are easier to consume by an autoregressive text interface.

A.8 Implementation Details

In this section, we present the implementation details of SemAE, covering the system prompt used for distillation, the computational cost analysis, and the hyperparameter settings.

Figure 8 illustrates the system prompt employed during the first training stage. This prompt is explicitly designed to be input into the VLM teacher model alongside camera images that feature projected robot trajectories. Its primary function is to strictly direct the VLM's attention towards the visualized motion path, thereby grounding abstract linguistic concepts into concrete spatial behaviors. By forcing the VLM to interpret the visual trajectory overlays, we guide the frozen VLM

System Prompt

You are a helpful robot assistant. You are given a task instruction in text and a trajectory of actions of a robot executing the task shown in the image, as highlighted color track line. The trajectory of actions shown in the image is a sequence of actions that the robot will execute in the following 2 seconds. The circles on the track line indicate where the grippers close at that location. You need to physically execute the trajectory of actions to complete the task in the real world.

Figure 8: **Prompt for motion primitive distillation.** The prompt specifies a two-second trajectory overlay and gripper-closing markers. A Q-Former extracts motion embeddings from the frozen VLM teacher's layer-wise hidden states to provide distillation targets for the SemAE encoder.

backbone to produce semantic embeddings that are not merely compressed action representations, but are semantically aligned with the teacher model's internal knowledge.

Table 9 provides a comprehensive analysis of the parameter size, computational operations (FLOPs), and inference latency for each individual module within SemAE. These metrics are critical for validating the model's viability for real-time, closed-loop robotic control. A key architectural advantage highlighted here is the computational asymmetry: while the VLM backbone is parameter-heavy, the Action Decoder is lightweight and efficient. This design allows the system to perform high-frequency action generation essential for reactive control, while the computationally intensive semantic reasoning can be decoupled or executed less frequently.

We provide an exhaustive listing of hyperparameters in Tables 10 to 17, covering configurations for four simulation benchmarks and three real-world robot setups. These tables detail critical training parameters including learning rates, batch sizes, optimizer settings, and architecture-specific dimensions tailored for each domain. For the LIBERO-PRO benchmark, we strictly adhere to the dataset configuration of the standard LIBERO benchmark to ensure a fair evaluation of long-horizon capabilities. In the case of the Polaris benchmark, which serves as a high-fidelity digital twin for the Droid robot, we implement a specific domain adaptation strategy. Following official protocols to bridge the real-to-sim gap, we mix the real-world Droid dataset with 10% simulation data in training.

Table 9: **Computational cost.** We report parameter counts, FLOPs, and module-level forward times for a single image and language prompt. Checkmarks in Train Time Only mark modules used exclusively during distillation. SemAE Decoder latency includes ten rectified-flow forward passes.

Model Part	Train Time Only	Parameters	FLOPS	Forward Time
Image Encoder (Qwen ViT (Bai et al., 2025))	×	668.7M	644.3G	27.6ms
LLM Backbone (Qwen 2.5 (Bai et al., 2025))	×	3754.6M	1391.6G	19.1ms
Q-Former	✓	453.3M	38.7G	19.8ms
SemAE Encoder	✓	558.4M	2.83G	60.9ms
SemAE Decoder	×	223.5M	0.02G	35.9ms
Distill Adapter	✓	97.2M	0.19G	7.1ms
Training Total	–	5755.8M	2077.6G	–
Inference Total	–	4646.9M	2035.9G	82.6ms

Table 10: **LIBERO hyperparameters.** Training and inference settings for SemAE on LIBERO, including optimization, encoder-decoder architecture, action chunk lengths, and action patch sizes.

Parameters	**Value**
Distillation Loss Weight λ_{dist}	1.0
Action Reconstruction Loss Weight λ_{recon}	1.0
Distillation Temperature β	2
Optimizer	AdamW
Learning Rate Scheduler	Cosine
Vision Tower Learning Rate	2e-5
Vision Merger Learning Rate	1e-4
Base Learning Rate	1e-4
Q-Former Learning Rate	1e-4
Encoder Learning Rate	5e-5
Decoder Learning Rate	1e-4
Global Blocks Learning Rate	5e-5
Adapter Learning Rate	2e-4
Training Data	LIBERO
Canera View	Static, Wirst
Image Size	256x256
Warmup Steps	1000
Total Training Epochs	10
Stage 1 Warmup Ratio	0.1
Stage 2 Warmup Ratio	0.03
Mix Precision	True
Stage 1 Per Device Batch Size	128
Stage 2 Per Device Batch Size	64
Gradient Accumulation Steps	1
Gradient Checkpointing	True
GPU Number	8 A100
Q-Former Hidden Size	2048
Q-Former Intermediate Size	3072
Q-Former Attention Heads	16
Q-Former Hidden Layers	12
Q-Former Position Embedding Type	absolute
VLM Backbone Hidden Size	2048
Encoder Hidden Size	1280
Encoder Transformer Layers	26
Encoder Attention Heads	16
Intermediate Size	3420
Hidden Activation	GELU
Normalization Type	RMS Norm
Layer Norm Epsilon	1e-6
Latent Channels	64
Global Blocks Depth	6
Dropout	0.0
Positional Encoding Type	Sinusoidal
Learnable Positional Embedding	False
Conv Upscale	False
Global Blocks Depth	6
Decoder Layers	12
Noise Concatenation	False
Sampling Steps	100
Training Schedule	Fat Lognormal
Training Action Length	16
Executable Action Length	8
Action Dimension	7
State Dimension	8
Encoder Channels	64
Encoder Channel Multipliers	(1, 2, 4)
Encoder Conv Block Nums	3
Action Patch Size	2

Table 11: **Bridge V2 hyperparameters.** SemAE settings for training on Bridge V2 and evaluation in SimplerEnv, including optimization, model architecture, action chunk lengths, and patch sizes.

Parameters	**Value**
Distillation Loss Weight λ_{dist}	1.0
Action Reconstruction Loss Weight λ_{recon}	1.0
Distillation Temperature β	2
Optimizer	AdamW
Learning Rate Scheduler	Cosine
Vision Tower Learning Rate	2e-5
Vision Merger Learning Rate	1e-4
Base Learning Rate	1e-4
Q-Former Learning Rate	1e-4
Encoder Learning Rate	5e-5
Decoder Learning Rate	1e-4
Global Blocks Learning Rate	5e-5
Adapter Learning Rate	2e-4
Training Data	Bridge V2
Canera View	Static
Image Size	256x256
Warmup Steps	1000
Total Training Epochs	5
Stage 1 Warmup Ratio	0.1
Stage 2 Warmup Ratio	0.03
Mix Precision	True
Stage 1 Per Device Batch Size	128
Stage 2 Per Device Batch Size	64
Gradient Accumulation Steps	1
Gradient Checkpointing	True
GPU Number	8 A100
Q-Former Hidden Size	2048
Q-Former Intermediate Size	3072
Q-Former Attention Heads	16
Q-Former Hidden Layers	12
Q-Former Position Embedding Type	absolute
VLM Backbone Hidden Size	2048
Encoder Hidden Size	1280
Encoder Transformer Layers	26
Encoder Attention Heads	16
Intermediate Size	3420
Hidden Activation	GELU
Normalization Type	RMS Norm
Layer Norm Epsilon	1e-6
Latent Channels	64
Global Blocks Depth	6
Dropout	0.0
Positional Encoding Type	Sinusoidal
Learnable Positional Embedding	False
Conv Upscale	False
Global Blocks Depth	6
Decoder Layers	12
Noise Concatenation	False
Sampling Steps	100
Training Schedule	Fat Lognormal
Training Action Length	8
Executable Action Length	4
Action Dimension	7
State Dimension	8
Encoder Channels	64
Encoder Channel Multipliers	(1, 2, 4)
Encoder Conv Block Nums	3
Action Patch Size	1

Table 12: **Fractal hyperparameters.** SemAE settings for training on Fractal and evaluation in SimplerEnv, including optimization, model architecture, action chunk lengths, and action patch sizes.

Parameters	**Value**
Distillation Loss Weight λ_{dist}	1.0
Action Reconstruction Loss Weight λ_{recon}	1.0
Distillation Temperature β	2
Optimizer	AdamW
Learning Rate Scheduler	Cosine
Vision Tower Learning Rate	2e-5
Vision Merger Learning Rate	1e-4
Base Learning Rate	1e-4
Q-Former Learning Rate	1e-4
Encoder Learning Rate	2e-5
Decoder Learning Rate	1e-4
Global Blocks Learning Rate	5e-5
Adapter Learning Rate	2e-4
Training Data	Fractal
Canera View	Static
Image Size	256x256
Warmup Steps	1000
Total Training Epochs	5
Stage 1 Warmup Ratio	0.1
Stage 2 Warmup Ratio	0.03
Mix Precision	True
Stage 1 Per Device Batch Size	128
Stage 2 Per Device Batch Size	64
Gradient Accumulation Steps	1
Gradient Checkpointing	True
GPU Number	8 A100
Q-Former Hidden Size	2048
Q-Former Intermediate Size	3072
Q-Former Attention Heads	16
Q-Former Hidden Layers	12
Q-Former Position Embedding Type	absolute
VLM Backbone Hidden Size	2048
Encoder Hidden Size	1280
Encoder Transformer Layers	26
Encoder Attention Heads	16
Intermediate Size	3420
Hidden Activation	GELU
Normalization Type	RMS Norm
Layer Norm Epsilon	1e-6
Latent Channels	64
Global Blocks Depth	6
Dropout	0.0
Positional Encoding Type	Sinusoidal
Learnable Positional Embedding	False
Conv Upscale	False
Global Blocks Depth	6
Decoder Layers	12
Noise Concatenation	False
Sampling Steps	100
Training Schedule	Fat Lognormal
Training Action Length	8
Executable Action Length	4
Action Dimension	7
State Dimension	8
Encoder Channels	64
Encoder Channel Multipliers	(1, 2)
Encoder Conv Block Nums	2
Action Patch Size	2

Table 13: **LIBERO-PRO hyperparameters.** SemAE is trained on LIBERO for evaluation on LIBERO-PRO. We list optimization, model architecture, action chunk lengths, and patch sizes.

Parameters	**Value**
Distillation Loss Weight λ_{dist}	1.0
Action Reconstruction Loss Weight λ_{recon}	1.0
Distillation Temperature β	2
Optimizer	AdamW
Learning Rate Scheduler	Cosine
Vision Tower Learning Rate	2e-5
Vision Merger Learning Rate	1e-4
Base Learning Rate	1e-4
Q-Former Learning Rate	1e-4
Encoder Learning Rate	5e-5
Decoder Learning Rate	1e-4
Global Blocks Learning Rate	5e-5
Adapter Learning Rate	2e-4
Training Data	LIBERO
Canera View	Static, Wirst
Image Size	256x256
Warmup Steps	1000
Total Training Epochs	10
Stage 1 Warmup Ratio	0.1
Stage 2 Warmup Ratio	0.03
Mix Precision	True
Stage 1 Per Device Batch Size	128
Stage 2 Per Device Batch Size	64
Gradient Accumulation Steps	1
Gradient Checkpointing	True
GPU Number	8 A100
Q-Former Hidden Size	2048
Q-Former Intermediate Size	3072
Q-Former Attention Heads	16
Q-Former Hidden Layers	12
Q-Former Position Embedding Type	absolute
VLM Backbone Hidden Size	2048
Encoder Hidden Size	1280
Encoder Transformer Layers	26
Encoder Attention Heads	16
Intermediate Size	3420
Hidden Activation	GELU
Normalization Type	RMS Norm
Layer Norm Epsilon	1e-6
Latent Channels	64
Global Blocks Depth	6
Dropout	0.0
Positional Encoding Type	Sinusoidal
Learnable Positional Embedding	False
Conv Upscale	False
Global Blocks Depth	6
Decoder Layers	12
Noise Concatenation	False
Sampling Steps	100
Training Schedule	Fat Lognormal
Training Action Length	16
Executable Action Length	8
Action Dimension	7
State Dimension	8
Encoder Channels	64
Encoder Channel Multipliers	(1, 2, 4)
Encoder Conv Block Nums	3
Action Patch Size	2

Table 14: **Polaris hyperparameters.** SemAE settings for training on DROID and Polaris data, including optimization, encoder-decoder architecture, action chunk lengths, and action patch sizes.

Parameters	Value
Distillation Loss Weight λ_{dist}	1.0
Action Reconstruction Loss Weight λ_{recon}	1.0
Distillation Temperature β	2
Optimizer	AdamW
Learning Rate Scheduler	Cosine
Vision Tower Learning Rate	2e-5
Vision Merger Learning Rate	1e-4
Base Learning Rate	1e-4
Q-Former Learning Rate	1e-4
Encoder Learning Rate	2e-5
Decoder Learning Rate	1e-4
Global Blocks Learning Rate	2e-5
Adapter Learning Rate	2e-4
Training Data	Droid, Polaris
Canera View	Static, Wirst
Image Size	256x256
Warmup Steps	1000
Total Training Epochs	12
Stage 1 Warmup Ratio	0.1
Stage 2 Warmup Ratio	0.03
Mix Precision	True
Stage 1 Per Device Batch Size	128
Stage 2 Per Device Batch Size	64
Gradient Accumulation Steps	1
Gradient Checkpointing	True
GPU Number	8 A100
Q-Former Hidden Size	2048
Q-Former Intermediate Size	3072
Q-Former Attention Heads	16
Q-Former Hidden Layers	12
Q-Former Position Embedding Type	absolute
VLM Backbone Hidden Size	2048
Encoder Hidden Size	1280
Encoder Transformer Layers	26
Encoder Attention Heads	16
Intermediate Size	3420
Hidden Activation	GELU
Normalization Type	RMS Norm
Layer Norm Epsilon	1e-6
Latent Channels	64
Global Blocks Depth	6
Dropout	0.0
Positional Encoding Type	Sinusoidal
Learnable Positional Embedding	False
Conv Upscale	False
Global Blocks Depth	6
Decoder Layers	12
Noise Concatenation	False
Sampling Steps	100
Training Schedule	Fat Lognormal
Training Action Length	32
Executable Action Length	8
Action Dimension	7
State Dimension	8
Encoder Channels	64
Encoder Channel Multipliers	(1, 2, 4)
Encoder Conv Block Nums	3
Action Patch Size	4

Table 15: **DROID hyperparameters.** SemAE settings for training on DROID and real-world evaluation, including optimization, model architecture, action chunk lengths, and action patch sizes.

Parameters	**Value**
Distillation Loss Weight λ_{dist}	1.0
Action Reconstruction Loss Weight λ_{recon}	0.8
Distillation Temperature β	2.1
Optimizer	AdamW
Learning Rate Scheduler	Cosine
Vision Tower Learning Rate	2e-5
Vision Merger Learning Rate	1e-4
Base Learning Rate	1e-4
Q-Former Learning Rate	1e-4
Encoder Learning Rate	2e-5
Decoder Learning Rate	1e-4
Global Blocks Learning Rate	2e-5
Adapter Learning Rate	2e-4
Training Data	Droid
Canera View	Left, Wirst
Image Size	256x256
Warmup Steps	1000
Total Training Epochs	10
Stage 1 Warmup Ratio	0.1
Stage 2 Warmup Ratio	0.03
Mix Precision	True
Stage 1 Per Device Batch Size	128
Stage 2 Per Device Batch Size	64
Gradient Accumulation Steps	1
Gradient Checkpointing	True
GPU Number	4 A100
Q-Former Hidden Size	2048
Q-Former Intermediate Size	3072
Q-Former Attention Heads	16
Q-Former Hidden Layers	12
Q-Former Position Embedding Type	absolute
VLM Backbone Hidden Size	2048
Encoder Hidden Size	1280
Encoder Transformer Layers	26
Encoder Attention Heads	16
Intermediate Size	3420
Hidden Activation	GELU
Normalization Type	RMS Norm
Layer Norm Epsilon	1e-6
Latent Channels	64
Global Blocks Depth	6
Dropout	0.0
Positional Encoding Type	Sinusoidal
Learnable Positional Embedding	False
Conv Upscale	False
Global Blocks Depth	6
Decoder Layers	12
Noise Concatenation	False
Sampling Steps	100
Training Schedule	Fat Lognormal
Training Action Length	32
Executable Action Length	8
Action Dimension	7
State Dimension	8
Encoder Channels	64
Encoder Channel Multipliers	(1, 2, 4)
Encoder Conv Block Nums	3
Action Patch Size	4

Table 16: **Franka hyperparameters.** Training and inference settings for real-world Franka experiments, including optimization, encoder-decoder architecture, action chunk lengths, and patch sizes.

Parameters	Value
Distillation Loss Weight λ_{dist}	1.0
Action Reconstruction Loss Weight λ_{recon}	1.0
Distillation Temperature β	2.0
Optimizer	AdamW
Learning Rate Scheduler	Cosine
Vision Tower Learning Rate	2e-5
Vision Merger Learning Rate	1e-4
Base Learning Rate	1e-4
Q-Former Learning Rate	1e-4
Encoder Learning Rate	2e-5
Decoder Learning Rate	1e-4
Global Blocks Learning Rate	2e-5
Adapter Learning Rate	2e-4
Training Data	Franka
Canera View	Static, Wirst
Image Size	256x256
Warmup Steps	1000
Total Training Epochs	5
Stage 1 Warmup Ratio	0.1
Stage 2 Warmup Ratio	0.03
Mix Precision	True
Stage 1 Per Device Batch Size	128
Stage 2 Per Device Batch Size	64
Gradient Accumulation Steps	1
Gradient Checkpointing	True
GPU Number	8 A100
Q-Former Hidden Size	2048
Q-Former Intermediate Size	3072
Q-Former Attention Heads	16
Q-Former Hidden Layers	12
Q-Former Position Embedding Type	absolute
VLM Backbone Hidden Size	2048
Encoder Hidden Size	1280
Encoder Transformer Layers	26
Encoder Attention Heads	16
Intermediate Size	3420
Hidden Activation	GELU
Normalization Type	RMS Norm
Layer Norm Epsilon	1e-6
Latent Channels	64
Global Blocks Depth	6
Dropout	0.0
Positional Encoding Type	Sinusoidal
Learnable Positional Embedding	False
Conv Upscale	False
Global Blocks Depth	6
Decoder Layers	12
Noise Concatenation	False
Sampling Steps	100
Training Schedule	Fat Lognormal
Training Action Length	40
Executable Action Length	10
Action Dimension	7
State Dimension	8
Encoder Channels	64
Encoder Channel Multipliers	(1, 2, 4)
Encoder Conv Block Nums	3
Action Patch Size	5

Table 17: **WidowX hyperparameters.** Settings for Bridge V2 training and real-world WidowX evaluation, including optimization, model architecture, action chunk lengths, and patch sizes.

Parameters	**Value**
Distillation Loss Weight λ_{dist}	1.0
Action Reconstruction Loss Weight λ_{recon}	1.0
Distillation Temperature β	2.0
Optimizer	AdamW
Learning Rate Scheduler	Cosine
Vision Tower Learning Rate	2e-5
Vision Merger Learning Rate	1e-4
Base Learning Rate	1e-4
Q-Former Learning Rate	1e-4
Encoder Learning Rate	2e-5
Decoder Learning Rate	1e-4
Global Blocks Learning Rate	2e-5
Adapter Learning Rate	2e-4
Training Data	Bridge V2
Canera View	Static
Image Size	256x256
Warmup Steps	1000
Total Training Epochs	8
Stage 1 Warmup Ratio	0.1
Stage 2 Warmup Ratio	0.03
Mix Precision	True
Stage 1 Per Device Batch Size	128
Stage 2 Per Device Batch Size	64
Gradient Accumulation Steps	1
Gradient Checkpointing	True
GPU Number	4 A100
Q-Former Hidden Size	2048
Q-Former Intermediate Size	3072
Q-Former Attention Heads	16
Q-Former Hidden Layers	12
Q-Former Position Embedding Type	absolute
VLM Backbone Hidden Size	2048
Encoder Hidden Size	1280
Encoder Transformer Layers	26
Encoder Attention Heads	16
Intermediate Size	3420
Hidden Activation	GELU
Normalization Type	RMS Norm
Layer Norm Epsilon	1e-6
Latent Channels	64
Global Blocks Depth	6
Dropout	0.0
Positional Encoding Type	Sinusoidal
Learnable Positional Embedding	False
Conv Upscale	False
Global Blocks Depth	6
Decoder Layers	12
Noise Concatenation	False
Sampling Steps	100
Training Schedule	Fat Lognormal
Training Action Length	8
Executable Action Length	4
Action Dimension	7
State Dimension	8
Encoder Channels	64
Encoder Channel Multipliers	(1, 2, 4)
Encoder Conv Block Nums	3
Action Patch Size	1

A.9 Simulation Benchmark Results

We report the full simulation results for the benchmark protocols described in Appendix A.10.1, using the baseline policies listed in Appendix A.10.2. Tables 18 and 19 report task-suite performance on LIBERO and robustness to the distribution shifts in LIBERO-PRO. Table 20 reports SimplerEnv results for the WidowX and Google Robot settings, including text-action co-training. Table 21 reports per-task Polaris results and the component ablations analyzed in the main paper.

Table 18: **LIBERO benchmark results.** We report success rates (SR, %) and ranks across four task suites, each with 10 tasks and 50 evaluation rollouts per task (seed 7). Values after "±" denote standard errors (SE); "–" indicates unavailable errors or ranks. Ranks use point-estimate SR.

Model	LIBERO-Spatial		LIBERO-Object		LIBERO-Goal		LIBERO-Long		Average	
	SR	Rank	SR	Rank	SR	Rank	SR	Rank	SR	Rank
Diffusion Policy (Chi et al., 2023)	78.5 ± 1.1%	15	87.5 ± 0.7%	14	73.5 ± 1.2%	15	64.8 ± 1.3%	12	76.1 ± 0.7%	14
OpenVLA (Kim et al., 2024)	84.7 ± 0.9%	14	88.4 ± 0.8%	13	79.2 ± 1.0%	11	53.7 ± 1.3%	16	76.5 ± 0.6%	13
OpenVLA-OFT (Kim et al., 2025)	97.6 ± 0.9%	4	98.4 ± 0.8%	6	97.9 ± 1.0%	4	94.5 ± 1.3%	3	97.1 ± 0.6%	3
DexVLA (Wen et al., 2025)	97.2 ± 0.6%	5	99.1 ± 0.5%	3	95.6 ± 0.8%	6	84.7 ± 1.0%	10	94.2 ± 0.4%	6
OpenHelix (Cui et al., 2025)	88.9 ± 0.9%	12	91.7 ± 0.4%	11	74.6 ± 0.5%	14	69.1 ± 0.8%	11	81.1 ± 0.5%	11
UniVLA (Bu et al., 2025b)	96.5 ± –%	7	96.8 ± –%	10	95.6 ± –%	6	92.0 ± –%	5	95.2 ± –%	5
GR00T N1 (Bjorck et al., 2025)	94.4 ± 0.9%	9	97.6 ± 1.0%	8	93.0 ± 1.2%	9	90.6 ± 1.0%	6	93.9 ± 1.1%	7
π_0 (Black et al., 2024)	96.8 ± 0.8%	6	98.8 ± 0.9%	4	95.8 ± 1.1%	5	85.2 ± 1.2%	9	94.2 ± 0.9%	6
π_0-FAST (Pertsch et al., 2025)	96.4 ± 0.7%	8	96.8 ± 0.7%	10	88.6 ± 1.0%	10	60.2 ± 1.4%	14	85.5 ± 1.0%	10
$\pi_{0.5}$ (Intelligence et al., 2025)	**98.8 ± 0.2%**	**1**	98.2 ± 0.3%	7	98.0 ± 0.9%	3	92.4 ± 0.6%	4	96.8 ± 0.7%	4
NORA (Hung et al., 2025b)	92.2 ± –%	–	95.4 ± –%	–	89.4 ± –%	–	74.6 ± –%	–	87.9 ± –%	–
NORA-1.5 (Hung et al., 2025a)	97.3 ± –%	–	96.4 ± –%	–	94.5 ± –%	–	89.6 ± –%	–	94.5 ± –%	–
SpatialVLA (Qu et al., 2025b)	88.2 ± 0.5%	13	89.9 ± 0.7%	12	78.6 ± 0.6%	12	55.5 ± 1.0%	15	78.1 ± 0.7%	12
FASTer (Liu et al., 2025)	98.0 ± –%	2	99.4 ± –%	2	98.6 ± –%	2	**95.4 ± –%**	**1**	97.9 ± –%	2
OmniSAT (Lyu et al., 2025)	94.1 ± –%	10	98.7 ± –%	5	94.6 ± –%	7	86.0 ± –%	8	93.4 ± –%	8
BEAST (Zhou et al., 2025)	92.9 ± –%	11	97.5 ± –%	9	93.1 ± –%	8	86.4 ± –%	7	92.5 ± –%	9
VQ-VLA (Wang et al., 2025)	68.2 ± 0.3%	16	71.9 ± 0.1%	15	75.3 ± 0.9%	13	61.3 ± 0.4%	13	69.2 ± –%	15
SemAE	97.9 ± 0.2%	3	**99.5 ± 0.4%**	**1**	**99.7 ± 0.2%**	**1**	95.1 ± 0.5%	2	**98.1 ± 1.2%**	**1**

Table 19: **LIBERO-PRO benchmark results.** Success rates cover object (Obj), position (Pos), semantic (Sem), task (Task), and environment (Env) shifts across four task suites. Average summarizes all 20 settings, with the highest score in each column shown in bold. SemAE leads on average.

Model	LIBERO-Spatial					LIBERO-Object					LIBERO-Goal					LIBERO-Long					Average
	Obj	Pos	Sem	Task	Env	Obj	Pos	Sem	Task	Env	Obj	Pos	Sem	Task	Env	Obj	Pos	Sem	Task	Env	
OpenVLA (Kim et al., 2024)	0.97	0.00	0.97	0.00	0.89	**0.98**	0.00	**0.98**	0.00	0.00	0.96	0.00	**0.98**	0.00	**0.98**	0.81	0.00	**0.96**	0.00	**0.85**	0.52
VQ-VLA (Kim et al., 2024)	0.97	0.00	0.97	0.00	0.89	**0.98**	0.00	**0.98**	0.00	0.00	0.96	0.00	**0.98**	0.00	**0.98**	0.81	0.00	**0.96**	0.00	**0.85**	0.52
π_0-FAST (Kim et al., 2024)	0.97	0.00	0.97	0.00	0.89	**0.98**	0.00	**0.98**	0.00	0.00	0.96	0.00	**0.98**	0.00	**0.98**	0.81	0.00	**0.96**	0.00	**0.85**	0.52
π_0 (Black et al., 2024)	0.95	0.00	0.97	0.00	0.60	0.94	0.00	0.90	0.00	0.29	0.94	0.00	0.93	0.00	0.39	0.79	0.00	0.82	0.00	0.27	0.44
$\pi_{0.5}$ (Intelligence et al., 2025)	0.97	**0.20**	0.97	0.01	0.46	**0.98**	**0.17**	0.96	0.01	**0.73**	0.97	**0.38**	0.97	0.00	0.46	0.92	0.08	0.93	0.01	0.46	0.53
SemAE	**0.98**	0.11	**0.98**	**0.12**	**0.92**	0.97	0.09	**0.98**	**0.12**	0.63	**0.98**	0.10	**0.98**	**0.11**	0.54	**0.93**	**0.21**	**0.96**	**0.37**	0.64	**0.59**

Table 20: **SimplerEnv benchmark results.** Success rates (%) are reported for four WidowX tasks and four Google Robot tasks under Visual Matching and Variant Aggregation. We compare baselines with SemAE and its interleaved text-action co-training variant; "–" denotes unavailable results.

Model	WidowX Benchmark				Google Robot (Matching)				Google Robot (Aggregation)				Average
	Spoon	Carrot	Stack	Eggplant	Coke	Move	Drawer	Apple	Coke	Move	Drawer	Apple	
RT-1 (Brohan et al., 2022)	–	–	–	–	85.7%	44.2%	**73.0%**	6.5%	89.8%	50.0%	32.3%	2.6%	48.0%
RT-1-X (Collaboration et al., 2024)	0.0%	4.2%	0.0%	0.0%	56.7%	31.7%	59.7%	40.7%	49.0%	32.3%	29.4%	10.1%	26.1%
RT-2-X (Collaboration et al., 2024)	–	–	–	–	78.7%	77.9%	25.0%	7.4%	82.3%	79.2%	35.3%	20.6%	50.8%
Octo-Small (Octo Model Team et al., 2024)	47.2%	9.7%	4.2%	56.9%	–	–	–	–	–	–	–	–	29.5%
Octo-Base (Octo Model Team et al., 2024)	12.5%	8.3%	0.0%	43.1%	17.0%	4.2%	22.7%	0.0%	0.6%	3.1%	1.1%	0.0%	9.3%
OpenVLA (Kim et al., 2024)	40.2%	20.4%	35.5%	54.6%	59.7%	46.2%	35.6%	0.0%	54.5%	47.7%	17.7%	0.0%	34.3%
OpenVLA-OFT (Kim et al., 2025)	12.5%	4.2%	8.3%	0.0%	–	–	–	–	–	–	–	–	6.2%
BEAST (Zhou et al., 2025)	41.7%	25.0%	20.8%	75.0%	–	–	–	–	–	–	–	–	40.6%
OmniSAT (Lyu et al., 2025)	58.3%	37.5%	29.2%	95.8%	–	–	–	–	–	–	–	–	55.2%
VQ-VLA (Wang et al., 2025)	12.5%	8.0%	6.0%	0.0%	–	–	–	–	–	–	–	–	6.6%
HPT (Wang et al., 2024)	–	–	–	–	56.0%	60.0%	24.0%	–	–	–	–	–	46.6%
GR00T N1.5 (Bjorck et al., 2025)	–	–	–	–	69.3%	68.7%	35.8%	–	46.7%	62.9%	17.5%	–	50.1%
RoboVLM (Li et al., 2024c)	45.8%	20.8%	4.2%	79.2%	–	–	–	–	–	–	–	–	37.5%
SpatialVLA (Qu et al., 2025b)	20.8%	20.8%	25.0%	70.8%	86.0%	77.9%	57.4%	0.0%	88.0%	72.7%	41.8%	6.3%	47.2%
TraceVLA (Zheng et al., 2024)	–	–	–	–	28.0%	53.7%	57.0%	0.0%	60.0%	56.4%	31.0%	0.0%	35.7%
UniVLA (Bu et al., 2025b)	54.2%	66.7%	50.0%	4.2%	–	–	–	–	–	–	–	–	43.7%
MiniVLA (Belkhale & Sadigh, 2024)	68.0%	44.0%	70.0%	14.0%	–	–	–	–	–	–	–	–	49.0%
NORA (Hung et al., 2025b)	–	–	–	–	74.2%	75.0%	31.7%	–	36.0%	73.0%	16.9%	–	51.1%
NORA-1.5 (Hung et al., 2025a)	–	–	–	–	92.8%	78.7%	62.2%	–	**95.0%**	75.7%	41.5%	–	74.3%
π_0 (Black et al., 2024)	83.8%	52.5%	52.5%	87.9%	97.9%	78.7%	62.2%	46.6%	90.1%	80.7%	27.6%	20.5%	65.0%
π_0-FAST (Pertsch et al., 2025)	77.5%	88.3%	68.3%	71.7%	75.3%	67.5%	42.9%	0.0%	77.6%	68.2%	31.3%	0.0%	55.7%
FASTer (Liu et al., 2025)	**91.7%**	**93.3%**	67.5%	**99.2%**	–	–	–	–	–	–	–	–	87.9%
SemAE	88.7%	89.2%	**79.8%**	94.3%	**98.2%**	81.2%	67.9%	49.3%	91.7%	82.1%	54.3%	22.4%	74.9%
SemAE + Interleave	89.3%	91.3%	78.3%	95.8%	**98.2%**	**83.7%**	72.5%	**54.1%**	93.2%	**84.6%**	**56.2%**	**23.6%**	76.7%

Table 21: **Polaris benchmark results.** Success rates and averages on six tasks compare baselines with SemAE and variants without semantic distillation, action flow decoding, or parallel decoding.

Model	BlockStack Kitchen	Food Bussing	Pan Clean	Move LatteCup	Organize Tools	TapeInto Container	Average
Paligemma Binning (Jain et al., 2025)	0.01	0.02	0.13	0.00	0.00	0.00	0.027
π_0-FAST (Pertsch et al., 2025)	0.26	0.54	0.34	0.35	0.35	0.55	0.398
π_0 (Black et al., 2024)	0.33	0.38	0.35	0.28	0.16	0.48	0.330
$\pi_{0.5}$ (Intelligence et al., 2025)	0.55	**0.59**	**0.57**	0.38	0.61	0.82	0.587
OpenVLA (Kim et al., 2024)	0.07	0.11	0.21	0.02	0.00	0.07	0.080
SemAE	**0.57**	0.54	**0.57**	**0.44**	**0.67**	**0.85**	**0.607**
SemAE w/o semantic distillation	0.37	0.38	0.55	0.27	0.51	0.84	0.487
SemAE w/o action flow decoding	0.51	0.42	0.47	0.41	0.63	0.80	0.540
SemAE w/o parallel decoding	**0.57**	0.50	0.49	**0.44**	0.60	0.77	0.562

A.10 Evaluation Protocol

Table 22: **Real-world DROID results.** Success rates are reported for seven tasks across Kitchen, Dining Table, and Cube scenes. The tasks test instruction following amid object distractors and changing spatial arrangements; Average summarizes model performance across all seven tasks.

Model	Kitchen			Dining Table			Cube	Average
	Flip Bottle	Add Ingredients	Cover the pot	Put Spoon	Put Bowl	Pick Bread	Arrange Cube	
π_0 (Black et al., 2024)	0.51	0.52	0.83	0.79	0.80	0.81	0.53	0.68
π_0-FAST (Pertsch et al., 2025)	0.48	0.69	0.85	0.83	0.89	0.87	0.67	0.75
$\pi_{0.5}$ (Intelligence et al., 2025)	0.59	0.69	**0.87**	0.81	**0.91**	0.85	0.65	0.77
OpenVLA-OFT (Kim et al., 2025)	0.33	0.57	0.79	0.61	0.75	0.76	0.31	0.59
GR00T N1.5 (Bjorck et al., 2025)	0.31	0.52	0.80	0.67	0.77	0.74	0.48	0.61
SemAE	**0.63**	**0.73**	**0.87**	**0.84**	0.89	**0.97**	**0.72**	**0.81**

A.10.1 Simulation Benchmarks

We evaluate four complementary benchmarks. For LIBERO and LIBERO-PRO, each method is fine-tuned on the corresponding task-suite demonstrations. SimplerEnv and Polaris evaluate generalization to visual and scene variations under fixed task definitions.

LIBERO (Liu et al., 2023) evaluates lifelong learning capabilities across four task suites: Spatial, Object, Goal, and Long. Each suite consists of 10 tasks that test spatial relations, object categories, goal specification, or long-horizon execution. Following OpenVLA (Kim et al., 2024), we conduct 50 evaluation rollouts for each task using a fixed seed (seed=7) and report the average success rate in Table 18.

SimplerEnv (Li et al., 2024d) assesses real-to-sim transfer performance by replicating Google Robot and WidowX setups with high visual fidelity. It evaluates policy robustness under Visual Matching and Variant Aggregation protocols, which introduce appearance and camera variations. We report per-task and average success rates for both robot setups in Table 20.

Polaris (Jain et al., 2025) is a high-fidelity benchmark aligned with the DROID platform, utilizing 3D Gaussian Splatting for realistic environment reconstruction. It evaluates policies on 6 manipulation tasks that mirror real-world scenarios. We perform 50 evaluation rollouts per task and report per-task and average success rates in Table 21.

LIBERO-PRO extends LIBERO to evaluate robustness against systematic perturbations. It introduces distribution shifts across five dimensions: object instances, initial positions, semantic instructions, task structures, and environmental conditions. We evaluate the policy for 50 episodes per task variant and report average success rates across suites and generalization settings in Table 19.

A.10.2 Baseline Policies

We compare our SemAE-based policy, denoted as SemAE in the tables, with representative vision-language-action models and generalist manipulation policies. These include OpenVLA and OpenVLA-OFT (Kim et al., 2024; 2025); π_0, π_0-FAST, and $\pi_{0.5}$ (Black et al., 2024; Pertsch et al., 2025; Intelligence et al., 2025); and, where applicable, SpatialVLA, FASTer, UniVLA, DexVLA, OpenHelix, GR00T, and Diffusion Policy (Qu et al., 2025b; Liu et al., 2025; Bu et al., 2025b; Wen et al., 2025; Cui et al., 2025; Bjorck et al., 2025; Chi et al., 2023).

A.10.3 Real-World Robots

The Franka and WidowX 250s evaluations report task success rates, intermediate grasp success where applicable, and overall averages in Tables 1 and 2. The DROID evaluation reports per-task success in Table 22.

WidowX 250s.

As shown in Figure 9, the detailed task specifications on WidowX 250s are: **Put eggplant in the basket.** A complex task requiring the robot to identify and pick an eggplant from a sink containing multiple vegetables, then place it in a yellow basket. This task evaluates object discrimination and spatial awareness.

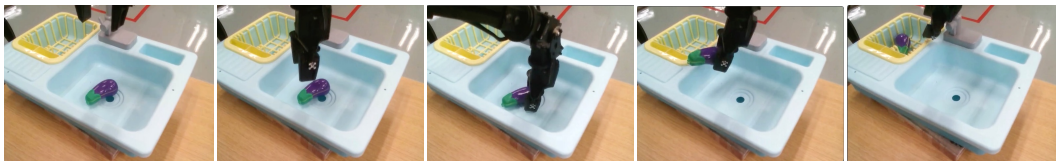
Put carrot on the plate. The robot needs to perform a pick-and-place task by grasping a carrot from the sink and placing it on a plate, assessing both grasping precision and placement accuracy.

Close microwave. The robot must close a toy microwave door positioned at various angles (30°, 45°, 60°, and 90°), testing the model's capability to manipulate articulated objects in different configurations.

Lift red pepper. A basic pick task requiring the robot to grasp and lift a red pepper from the sink, designed to evaluate the model's object localization accuracy.

Put green cup on the pink cloth. This task suite comprises two scenarios testing vertical spatial understanding. In the first scenario, the robot grasps a green cup positioned either on a stove or elevated on a yellow block. In the second scenario, the cup is placed either at the bottom of a sink or elevated on a bowl. This variation in object heights challenges the model's ability to adapt its manipulation strategy according to spatial configurations.

Put purple cup on the white plate. The robot must identify and transfer a purple cup to a white plate within a sink containing multiple plates, testing color recognition and precise manipulation.



Put eggplant in the basket: Put the eggplant in the basket.



Put carrot on the plate: Put the carrot on the plate.



Close microwave: Close the microwave.



Lift red pepper: Lift the red pepper.



Put green cup on the pink cloth : Put the green cup on the pink cloth.



Put purple cup on the white plate : Put the purple cup on the white plate.

Figure 9: **WidowX 250s manipulation sequences.** Six illustrated instructions cover vegetable pick-and-place, microwave closing, pepper lifting, and cup placement. Frames progress from left to right.

Franka. As shown in Figure 10, the detailed task specifications on Frank-Emika-Panda are: **Kitchen Banana.** A pick-and-place task where the robot must transfer a banana from the table to a basket. With only 50 human demonstrations, this task evaluates model performance with limited data.

Kitchen Pot. Another pick-and-place task requiring the robot to grasp a white bowl from the right side of the table and place it on a cutting board, trained with 100 human demonstrations.

Tea. The robot must push a teapot handle from a perpendicular to a parallel position relative to the desktop, using its gripper tip. This task tests the manipulation of revolute joints and includes 50 human demonstrations.

Plush Toy. The robot must identify and grasp the nearest plush toy among two options and place it on a green car. To rigorously assess spatial understanding, we systematically vary the relative positions of the plush toys during testing.

Three Cube. An instruction-following task where the robot must identify and place a specifically colored cube (red, green, or blue) onto a green car. Training included 50 demonstrations for each color, totaling 150.

DROID. All tasks in the DROID scenarios are trained based on the public DROID dataset. To rigorously evaluate the model's language following capabilities in realistic, unstructured environments, we establish three distinct scenarios: Kitchen, Dining Table, and Cube, as illustrated in Figure 11. Unlike standard benchmarks that often feature isolated objects, our setups are intentionally cluttered with multiple manipulatable items. This design specifically targets the model's *Affordance Identification* capability, demanding that the policy suppress the urge to interact with salient but irrelevant objects and strictly adhere to the semantic constraints imposed by the instruction.

In the *Kitchen* scenario, we designed three tasks to evaluate precision and object discrimination in a crowded workspace. The *Flip Bottle* task challenges the model to identify a specific fallen bottle amidst other kitchenware and execute a reorientation maneuver to set it upright. The *Add Ingredients* task requires the robot to grasp a specific cup containing ingredients and pour them into a pot, testing the ability to distinguish the correct container from potential distractors. The *Cover the Pot* task involves placing a transparent lid onto the pot, which necessitates precise geometric alignment and spatial understanding. Collectively, these tasks test the model's ability to ground specific verbs (flip, pour, cover) to the correct objects while ignoring other graspable items on the table.

The *Dining Table* scenario focuses on resisting spurious temporal correlations and handling complex object relationships through three tasks: Put Spoon, Put Bowl, and Pick Bread. *Put Spoon* requires the robot to identify a small, metallic spoon on a stand and place it into a blue bowl, testing fine-grained perception. *Pick Bread* requires retrieving a slice of bread from a toaster and placing it onto a steak, distinguishing it from other food items. Crucially, the *Put Bowl* task serves as a stress test for temporal independence. Many VLA models trained on demonstration data capture statistical regularities, such as the sequence "place spoon in bowl $\rightarrow$ place bowl on plate." We evaluate the model's ability to break this prior when instructed to simply "put the bowl on the plate" while the spoon is still outside. Failures here, where baselines like OpenVLA-OFT and GR00T N1.5 hesitate or attempt to place the spoon first, indicate a reliance on memorized scripts over instruction following. SemAE's success here demonstrates its ability to decouple semantic instructions from learned temporal priors.

The final scenario, *Cube*, is explicitly designed to probe *Indirect Reference Handling* and robustness against spatial overfitting. We present three colored cubes and three corresponding cloths, with their positions randomized in every evaluation episode. The evaluation instruction is consistently "Place the cube on the cloth of the corresponding color." This setup forces the model to perform multi-step reasoning: it must first identify the color of the target cube and then visually search for the matching cloth, rather than relying on fixed spatial locations. This reveals whether a model truly understands the semantic concept of "correspondence" or merely memorizes "pick from location A, place at location B." The quantitative results, presented in Table 22, demonstrate SemAE's superior performance, confirming that its semantic-aligned action representations effectively generalize to dynamic spatial configurations and abstract relational instructions.

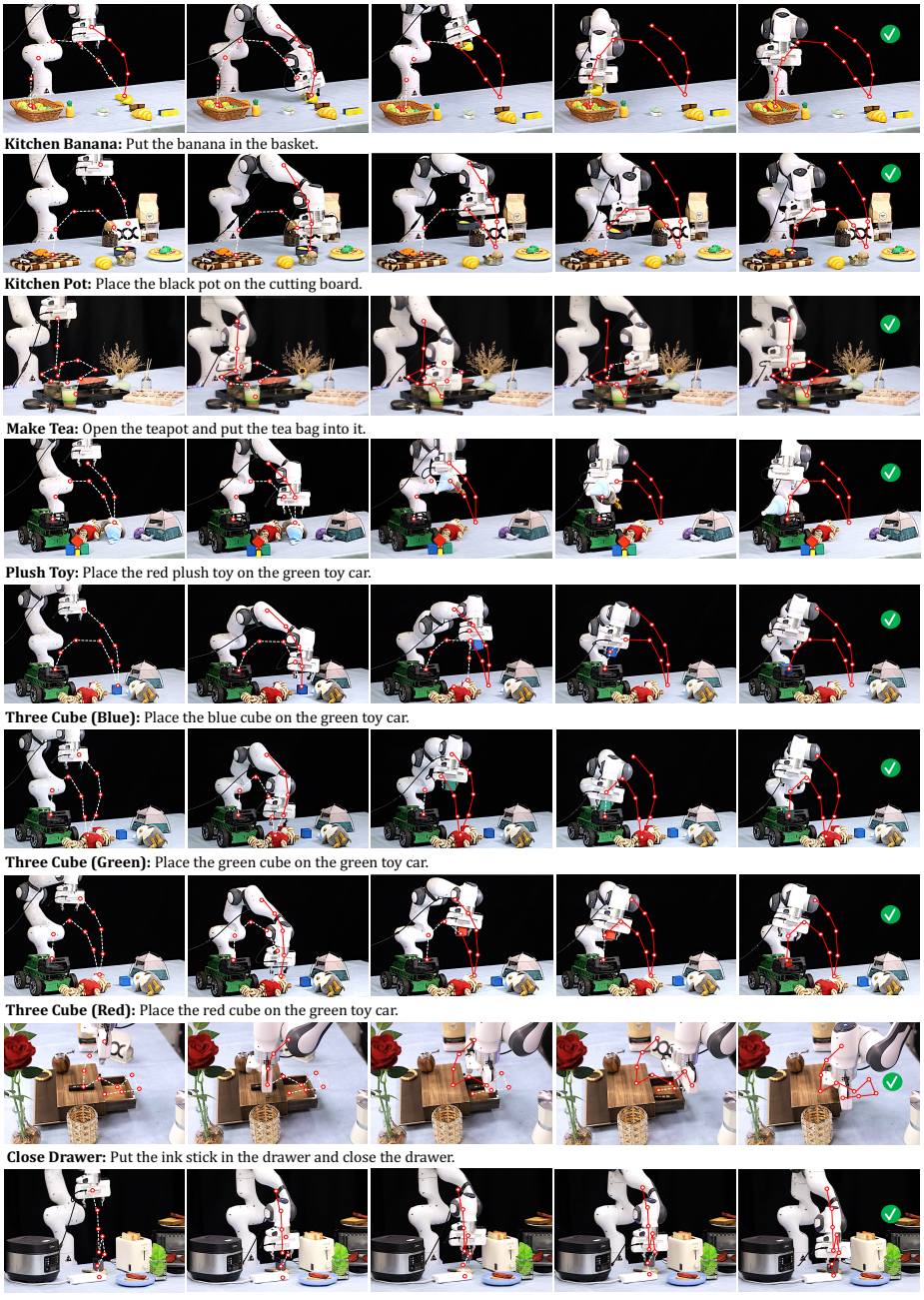


Figure 10: **Franka manipulation sequences.** Representative executions span kitchen pick-and-place, tea preparation, toy and colored-cube placement, drawer closing, and charger insertion. The three cube rows illustrate instructions for distinct colors; the frames progress from left to right.



Figure 11: **DROID manipulation sequences.** Seven tasks span the Cube, Kitchen, and Dining Table scenarios. The first three rows illustrate the same color-matching instruction with different cubes; subsequent rows show kitchen and dining tasks, with frames progressing from left to right.

B AI USE STATEMENT

We used OpenAI Codex during manuscript preparation and revision to translate author-provided descriptions, provide feedback on methodology and evaluation protocols, and assist with interpreting and checking reported results. Codex also assisted with drafting and editing text, reorganizing sections, migrating the manuscript to the ICLR LaTeX template, formatting and compilation checks, and writing a script to aggregate existing platform-level success rates and update the corresponding summary figure. The authors remain responsible for all content and artifacts in this paper, including text, claims, code, and figures produced with AI assistance.