DFA-VLA: DYNAMIC FINE-GRAINED ALIGNMENT FOR VISION-LANGUAGE-ACTION MODELS IN ROBOTIC MANIPULATION

Anonymous authors
Paper under double-blind review

ABSTRACT

With the rapid advancement of robotic hardware and software technologies, Vision-Language-Action (VLA) models have become pivotal for mapping multimodal inputs to continuous robot actions. However, mainstream VLA models often suffer from poor modeling of fine-grained visual elements (e.g., occluded regions and small objects) and an over-reliance on static cross-modal attention, restricting adaptability in complex, open environments. A plausible but underexamined failure mode is attention dispersion under visual clutter: when distractor objects are present, a dense visual prefix may spread cross-modal attention across irrelevant tokens and degrade manipulation precision even when the target object is clearly visible. To address these limitations, we propose DFA-VLA (Dynamic Fine-Grained Alignment for Vision-Language-Action Models), integrating two key components: (1) the Multi-scale Visual-Semantic Modeling (MVSM) module, which preserves global scene context while extracting segment-guided local object features; and (2) the Dynamic Fine-grained Alignment and Fusion (DFAF) module, which dynamically isolates task-critical object tokens via text-guided token retrieval and learnable gating before cross-modal fusion into a frozen language backbone. In real-world evaluations on 22 manipulation tasks (220 trials, scored 1, 0.5, or 0 for success, partial success, or failure), DFA-VLA trained for 10,000 steps (5,000 on LIBERO followed by 5,000 on BridgeData V2) obtains a mean score of 78.6% (95% Wilson interval [72.8%, 83.5%]) against 71.1% for the published OpenVLA policy, scoring higher on 16 of the 22 tasks, tying on 5, and lower on 1 (paired sign test $p < 10^{-3}$). In a paired 1,000-step clutter study on LIBERO-Spatial (32 scenes with 0/3/5/8 distractors), however, five-token prefixes, with or without segmentation, do not fit the demonstrations and reach only 3/32 and 1/32, while a LoRA-adapted dense control reaches 26/32 versus 22/32 for the published OpenVLA (exact McNemar $p = 0.29$). The two evaluations differ in budget, data, and scoring, and the physical evaluation has no matched dense or sparse control, so they are not directly comparable.

1 INTRODUCTION

In recent years, technological innovations in the field of artificial intelligence have laid a solid foundation for the vigorous development of embodied intelligence. From the rise of large language models (LLMs) based on the Transformer architecture, such as GPT-4 [1], BERT [2], and T5 [3], to the rapid evolution of vision-language models (VLMs), a series of breakthroughs have created the necessary conditions for the implementation of embodied intelligence.

Large Language Models (LLMs) support natural language processing (NLP) with their powerful semantic understanding and generation capabilities, while Vision-Language Models (VLMs) break down modal barriers through techniques such as cross-attention mechanisms and contrastive learning, promoting the development of cross-modal cognition. Against this backdrop, end-to-end vision-language-action models like OpenVLA [4], RT-2 [5], and PaLM-E [6] seek to enable robots to integrate environmental perception (via RGB or video), language understanding, and action execution into a unified framework. These unified VLA frameworks based on pre-trained VLMs or LLMs embody the fundamental workflow of embodied intelligence models for fixed robots, from high-level

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.

planning to low-level control, and establish a core paradigm for the technical implementation of embodied intelligence.

However, a dense visual prefix of hundreds of tokens gives the policy no explicit mechanism to focus on task-relevant regions when distractor objects are present. To address this, this paper proposes DFA-VLA, a segmentation-guided sparse visual prefix for OpenVLA: candidate global tokens and segmentation-derived local tokens are retrieved by the instruction and gated before they enter the language backbone. We ask a simple question: can a prefix of five tokens replace the dense 256-token prefix without losing manipulation performance under clutter? We answer it with matched controls (a LoRA dense variant, a LoRA sparse-global variant, and the published OpenVLA policy) and paired rollouts at four distractor counts, and we report the outcome as observed: under a 1000-step, single-seed fine-tuning budget, the dense control matches or slightly exceeds the published policy, whereas five-token prefixes collapse. We also report measured segmentation and pipeline cost and real-world results on physical robots.

2 Related Work

Classical Manipulation and Vision-Language-Action Models. Traditional robotic manipulation approaches relied on sampling-based motion planning (e.g., RRT/RRT* [7, 8]) for collision-free trajectory generation and rule-based systems for repetitive tasks. While effective in structured environments, these methods require precise goal specifications and cannot interpret semantic instructions, motivating the development of end-to-end vision-language-action (VLA) frameworks. Recent VLAs such as RT-2, Octo, and OpenVLA [5, 9, 4] align visual tokens with language representations to generate robotic actions, demonstrating real-world viability. However, their static cross-modal attention mechanisms tend to over-attend to irrelevant visual regions, struggle with dynamic modality balance adjustment, and exhibit limited generalization in cluttered scenes and long-horizon tasks. We address these limitations through segmentation-guided token retrieval and learnable gating that adaptively balances vision-language fusion based on task context.

Fine-Grained Visual Modeling. Multi-scale architectures (FPN, PANet [10, 11]) and vision transformers (Swin [12]) have advanced feature extraction, while segmentation models like SAM [13] provide precise object masks for localizing small or occluded objects. Yet prior work rarely couples such masks with dynamic cross-modal routing or task-adaptive computation allocation. Our MVSM module exploits DinoV2 [14] for global scene context alongside SAM-guided local regions, while our DFAF module retrieves and gates a small set of task-relevant tokens, addressing the efficiency-accuracy trade-off in fine-grained perception.

Sim2Real Transfer and Model Adaptation. Sim2Real transfer faces persistent challenges from visual domain gaps (rendering artifacts, sensor noise), dynamics mismatches (idealized friction, contact modeling), and multi-modal misalignments that impede generalization to long-horizon, semantically complex tasks [15, 16, 17]. Concurrently, parameter-efficient fine-tuning (PEFT) methods like LoRA [18, 19] enable task adaptation with minimal trainable parameters but suffer performance degradation under quantization, with modality-specific sensitivities (e.g., vision encoders being more sensitive than language decoders) complicating deployment [20, 21]. We evaluate our approach in LIBERO simulation and on physical robots, using LoRA-based adaptation of a pretrained VLA; we do not study quantized deployment.

3 Methodology

This section describes DFA-VLA as implemented: an adapter on the pretrained OpenVLA LIBERO-Spatial policy that replaces the dense 256-token visual prefix by a small set of retrieved and gated tokens drawn from global and segmentation-guided local candidates. We describe (1) candidate construction, (2) instruction-guided retrieval and gating, and (3) parameter-efficient training.

Model Architecture. DFA-VLA starts from the public OpenVLA checkpoint [4]: a fused DINOv2 [14] and SigLIP visual encoder with a projector, and a Llama 2 [22] decoder that emits discretized action tokens. The model takes one current 224×224 RGB frame (official camera orientation and center crop) and the natural language instruction, and outputs the standard 7-D OpenVLA action tokens (translation, rotation, gripper); the action vocabulary, dataset statistics, and LIBERO

gripper conversion are unchanged, and there is no continuous action head. The visual encoder, projector, and Llama 2 weights are frozen. Our additions are a Multi-scale Visual-Semantic Modeling (MVSM) module that constructs candidates, a Dynamic Fine-grained Alignment and Fusion (DFAF) module that retrieves and gates them, and rank-8 LoRA adapters. Figures 1 and 2 illustrate the two modules.

Multi-scale candidates (MVSM). The frozen fused encoder and projector give 256 global tokens of dimension 4096. MobileSAM automatic mask generation, a lightweight variant of SAM [13], supplies at most $M = 16$ class-agnostic candidate masks, ordered deterministically by predicted IoU, stability score, and box coordinates. For each valid mask, pixels outside the mask are filled with RGB value 127, the bounding box is cropped, and the crop passes through the same processor, encoder, and projector; mean pooling the projected crop patches gives one local token. If no valid mask remains, only global candidates are used. With global tokens g_i, local tokens l_j, and normalized boxes b_j, the candidate sequence is

$$V = [g_1, \ldots, g_{256}, l_1 + W_b b_1, \ldots, l_M + W_b b_M], \tag{1}$$

where W_b maps four box coordinates to 4096 dimensions and is initialized to zero; global tokens receive no second position encoding.

Retrieval and gating (DFAF). Let T be the OpenVLA input embeddings of the prompt and m a mask that excludes BOS, padding, and teacher-forced action tokens. The retrieval context and candidate scores are

$$t = \frac{\sum_l m_l T_l}{\sum_l m_l}, \qquad s_i = \cos(V_i, t). \tag{2}$$

The K candidates with the largest signed scores are selected ($K = 5$ by default) and restored to their original order before insertion. The mean input embedding is not a contextualized sentence representation, which limits sensitivity to word order and relations. Each selected token receives a scalar gate

$$\alpha_i = \sigma(V_i^\top w_v + t^\top w_t + c), \qquad F_i = \alpha_i V_i, \tag{3}$$

with the bias initialized so that the mean gate starts at ρ ($\rho = 0.9$ in the main setting). Retrieval itself is parameter-free. The prefix keeps the OpenVLA layout $[\mathrm{BOS}; F_1, \ldots, F_K;$ remaining prompt tokens$]$: the selected tokens take the place of the original visual prefix, and no decoder layer is replaced; the decoder is modified only by the LoRA adapters below. Algorithm 1 summarizes the procedure.

Training objective and parameters. The fused encoder, projector, and Llama 2 base weights are frozen. Rank-8 LoRA adapters are added to the q and v projections in all 32 decoder layers, and the box projection and gate parameters are trained (4,218,881 trainable parameters out of 7,545,456,065). Gradients flow through the frozen decoder to the prefix and adapters. The loss is $\mathcal{L} = \mathcal{L}_{\text{action}} + \lambda \mathcal{L}_{\text{gate}}$ with $\lambda = 0.01$, where $\mathcal{L}_{\text{action}}$ is the OpenVLA cross-entropy on action tokens and $\mathcal{L}_{\text{gate}} = \mathrm{mean}_{\text{batch}} |\mathrm{mean}_i \alpha_i - \rho|$. We do not use a contrastive alignment loss or a mask supervision loss.

Algorithm 1 Retrieval and gating (DFAF) of the visual prefix

Input: prompt embeddings T with mask m, candidates V, and $K = 5$
Output: gated prefix $F \in \mathbb{R}^{K \times 4096}$
Parameters: gate vectors w_v, w_t and bias c

// 1. Masked-mean instruction context
$t \leftarrow \sum_l m_l T_l / \sum_l m_l$
// 2. Signed-cosine retrieval of the top-K candidates
$s_i \leftarrow \cos(V_i, t)$ for every candidate i
Idx $\leftarrow \mathrm{TopK}(s, K)$, restored to the original candidate order
// 3. Scalar gating
$\alpha_i \leftarrow \sigma(V_i^\top w_v + t^\top w_t + c)$ for $i \in$ Idx
$F_i \leftarrow \alpha_i V_i$
return F, inserted after BOS in place of the visual prefix

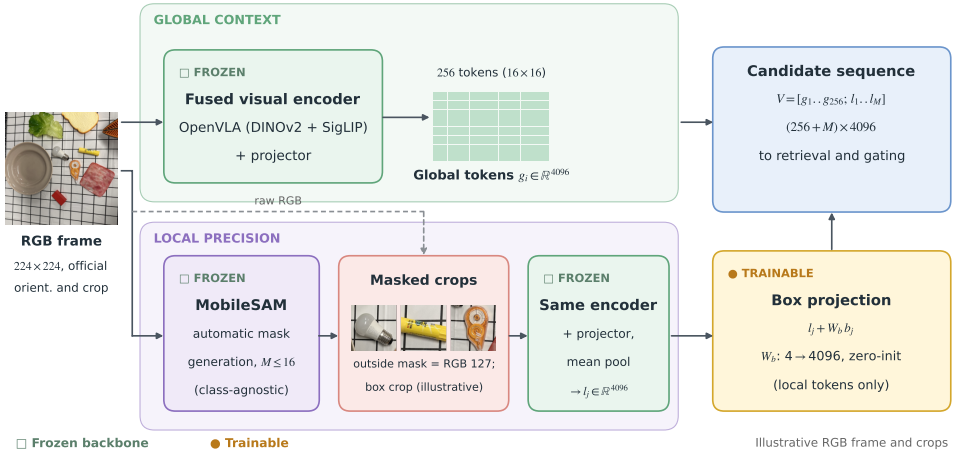


Figure 1: Candidate construction (MVSM). The full frame and MobileSAM local crops pass through the same frozen OpenVLA image processor, fused vision encoder, and projector; only local tokens receive an added box projection, which is initialized to zero.

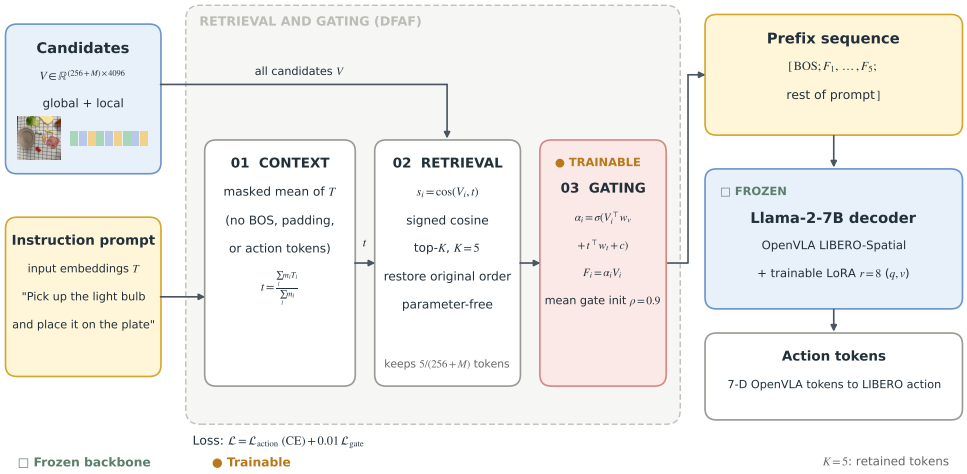


Figure 2: Retrieval and gating (DFAF). The masked-mean instruction context scores all candidates, the top-K are gated and placed at the original OpenVLA visual-prefix position, and the frozen decoder with LoRA adapters produces the original action tokens. The thumbnail and token strip in the candidate box are illustrative.

Action decoding. The decoder emits OpenVLA's discretized action tokens, which are de-tokenized with the checkpoint's dataset statistics into a 7-D end-effector command (translation, rotation, gripper), and the gripper is converted to the LIBERO convention.

4 EXPERIMENTS

Setting. We use the LIBERO-Spatial suite [23] and fine-tune from the official OpenVLA LIBERO-Spatial checkpoint [4] on the official demonstrations. All trained conditions use the same demonstrations, optimizer steps, LoRA rank and targets, action-token objective, and evaluation scenes: **LoRA dense** keeps all 256 global tokens; **LoRA sparse global** keeps the top-K of the 256 global tokens without segmentation; and **DFA-VLA** keeps the top-K of global and MobileSAM local candidates. The published OpenVLA checkpoint is an external reference.

Clutter scenes. Scenes receive 0, 3, 5, or 8 added objects. Each task, initial state, and seed shares one eight-object layout, and lower levels use nested prefixes of it. Scene files, state arrays, placement metadata, and hashes fix the pairing across policies and levels. The two episodes of a task have different initial states and independently sampled layouts, and the four levels of one episode share its initial state. Objects are anchored and collidable; a visual-only control was not run.

Executed pilot. Rollouts run on one NVIDIA A100 80 GB GPU, use LIBERO-Spatial, evaluation seed 7, a 220-step horizon after 10 settling steps, and the official OpenVLA image preprocessing, with two episodes per task and clutter level. The paired comparison covers tasks 0 to 3 (32 scenes per policy); the published policy was also rolled out on all ten tasks (80 scenes). Training uses one seed, 1000 optimizer steps, micro-batch size one with 16 gradient accumulation steps, AdamW with learning rate 2×10^{-4} and weight decay 0.01, $K = 5$, $\rho = 0.9$, and at most 16 MobileSAM candidates. We did not run the visual-only control, the other LIBERO suites, multiple training seeds, or ρ and K sweeps. With two episodes per task and level, every comparison is underpowered for small effects.

Statistics. For paired comparisons on identical scenes we report discordant counts and an exact McNemar test.

Real-world Validation. Physical validation is performed by deploying a fine-tuned policy directly onto industrial robotic platforms (Zhichang CR5 [25] and Lebai LM3 [26]) without collecting additional training data on these platforms, i.e., we evaluate transfer without platform-specific training. The real-world DFA-VLA has the same architecture and was trained for 10,000 optimizer steps in two stages, 5,000 steps on the LIBERO demonstrations followed by 5,000 steps on BridgeData V2 [24], a real-robot subset of Open X-Embodiment; no data collected on our own platforms was used. This is ten times the 1,000-step budget of the simulation pilot of Section 5.1. Physical tasks are chosen to be semantically analogous to LIBERO families and are adapted to the workspace and available objects. Quantitative and qualitative evaluation protocols for real-world tests are detailed in the appendix. Figure 4 shows the hardware setup.

5 RESULTS AND DISCUSSION

We evaluate DFA-VLA in a paired clutter study in simulation and on real-world robotic platforms; both are reported below.

5.1 TARGETED CLUTTER ANALYSIS

To isolate the effect of visual clutter, we procedurally augment LIBERO-Spatial scenes with increasing numbers of distractor objects (0, 3, 5, and 8 distractors) and evaluate four policies under an identical task and rollout protocol (32 paired scenes: tasks 0–3, 2 episodes per task and distractor level, i.e., 8 scenes per level): the published OpenVLA checkpoint, a LoRA fine-tuned dense control that keeps all 256 visual tokens, an unguided sparse-global control that keeps the top-K ($K = 5$) of the 256 global tokens without segmentation, and DFA-VLA. As reported in Table 1, published OpenVLA succeeds in 22/32 scenes and falls from 87.5% to 50.0% when distractor density reaches 8 objects. The dense control succeeds in 26/32 scenes; this difference is not statistically significant (exact McNemar test on paired scenes, 6 vs. 2 discordant scenes, $p = 0.29$). The sparse-global con-

Table 1: Targeted clutter analysis across distractor counts (32 paired scenes, 8 per level; success rate with successes/scenes). Mean degradation Δ is the mean over the four levels of the drop relative to the zero-distractor level.

Distractors	**OpenVLA**	**LoRA dense**	**Sparse global** ($K = 5$)	**DFA-VLA (1000 steps)**
0	87.5% (7/8)	87.5% (7/8)	0.0% (0/8)	12.5% (1/8)
3	62.5% (5/8)	87.5% (7/8)	12.5% (1/8)	0.0% (0/8)
5	75.0% (6/8)	75.0% (6/8)	25.0% (2/8)	0.0% (0/8)
8	50.0% (4/8)	75.0% (6/8)	0.0% (0/8)	0.0% (0/8)
Mean clutter	68.8% (22/32)	81.3% (26/32)	9.4% (3/32)	3.1% (1/32)
Mean Δ	-18.7%	-6.2%	—	-9.4%

trol collapses to 3/32 scenes ($p < 10^{-5}$ against both OpenVLA and the dense control), which suggests that, at this training budget, reducing the prefix to five tokens is harmful even before segmentation is added. DFA-VLA, which keeps five tokens selected from global and segmentation-derived candidates, succeeds in 1/32 scenes (3.1%, 95% Wilson interval [0.6%, 15.7%]; its only success is a zero-distractor scene). This is significantly worse than both the published OpenVLA ($p < 10^{-5}$) and the dense control ($p < 10^{-6}$), and statistically indistinguishable from the sparse-global control ($p = 0.62$). Under this 1,000-step, single-seed budget, segmentation-derived candidates therefore did not rescue the five-token prefix. The training logs are consistent with under-fitting rather than with an evaluation artifact: averaged over optimizer steps 901 to 1000, the training loss was 0.23 for the dense control (token accuracy 0.93), 2.12 for sparse-global (0.52), and 2.46 for DFA-VLA (0.47), so both five-token variants remained far from fitting the demonstrations. A plausible reading is that a rank-8 LoRA adapter trained for 1,000 steps cannot adapt the frozen decoder to a prefix of five tokens instead of the 256 it was pretrained with; we did not test this explanation with longer training or further ablations, so it remains a hypothesis. We note a non-monotonic artifact in OpenVLA between 3 distractors (62.5%) and 5 distractors (75.0%), which reflects finite sample variance across the $n = 8$ scenes per distractor level. All policies were trained for 1,000 steps with a single seed, so these pilot results should be read as exploratory.

Finally, we report the results of the robotic arm in real-world scenarios. Drawing on the classification logic of robotic manipulation tasks in the BridgeData V2 dataset, we divide the core capability verification tasks of OpenVLA into five major categories: visual generalization tasks, motion generalization tasks, physical generalization tasks, semantic generalization tasks, and language grounding tasks. Each category comprehensively tests the model's ability to adapt to real-world robotic operation scenarios from five dimensions: perception, motion, physical interaction, semantic understanding, and language-action mapping. Table 2 and its category summary (Table 4) give the per-task and per-category scores. Recomputed from all 220 trial-level scores, DFA-VLA obtains a mean score of 78.6% (95% Wilson interval [72.8%, 83.5%]) and the published OpenVLA policy 71.1% ([64.8%, 76.7%]), a 7.5-point difference whose intervals partly overlap. DFA-VLA scores higher than OpenVLA on 16 of the 22 tasks, ties on 5, and is lower on 1 (paired sign test excluding ties, $p = 2.7 \times 10^{-4}$; the tasks are not independent and each has only 10 trials). RT-2-X (52.7%), Octo (19.1%), and RT-1-X (18.9%) are lower. Because the physical evaluation contains no matched dense or sparse control, it does not isolate the effect of the sparse prefix. The specific categories and their task significances are as follows:

1) Visual Generalization Task: Focuses on the generalization of visual perception, verifying the model's ability to stably recognize target objects and scene information when unexpected changes occur in visual input. Its core value is to break free from the limitation of fixed visual samples, ensuring robots can stably complete perception tasks under dynamic visual conditions and lay the foundation for subsequent operations.

2) Motion Generalization Task: Targets the adaptability of action execution, ensuring the accuracy and safety of operations when action constraints and hardware configurations change. Its core value is to break through the limitations of single hardware and action templates, improve the versatility and safety of robot actions, and adapt to diverse hardware platforms and scene operation specifications.

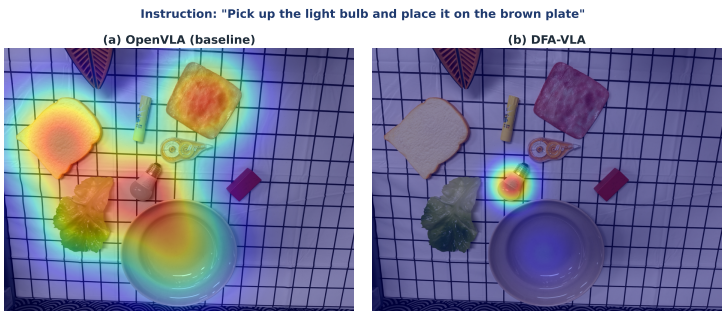


Figure 3: **Qualitative cross-modal attention maps for one example** (instruction: *"Pick up light bulb and place on brown plate"*), OpenVLA vs. DFA-VLA. Shown for illustration only; the maps are not used in any reported result and are not a quantitative analysis.

3) Physical Generalization Task: Revolves around the authenticity of physical interaction, enabling the model to understand and adapt to differences in object physical properties. Its core value is to establish the associated cognition of physical properties and operation strategies, making interaction conform to real physical laws and reducing operation errors caused by misjudgment of object physical properties.

4) Semantic Generalization Task: Focuses on the in-depth understanding of language commands, breaking through the limitation of fixed command templates to adapt to diverse and abstract semantic expressions. Its core value is to break the barrier between literal commands and deep semantics, realize natural human-robot language interaction, and adapt to scenarios requiring high-frequency natural communication like home services and medical assistance.

5) Language Grounding Task: Connects language commands and physical operations, accurately converting abstract language commands into specific robot actions. Its core value is to realize the key capability of language-driven robot operation, connect the entire link from human-robot interaction to actual execution, and serve as the core foundation for robot language control in scenarios like industrial sorting, warehouse handling, and home services.

Per-task results (22 tasks across 5 categories) are provided in Appendix A (Table 2). Failure modes are summarized approximately in Appendix A.3 (Table 5), and Figure 3 shows one illustrative attention example.

6 LIMITATIONS

This study has several limitations. The reliance on segmentation introduces an upstream perception cost: MobileSAM automatic mask generation takes 300.8 ms per frame (median, A100) and the complete DFA-VLA pipeline takes 3.3 s per step (about 0.3 Hz; Table 3), which is far from real-time control; we did not implement asynchronous proposal propagation. The simulation pilot used a 1,000-step budget, one tenth of the 10,000 steps used for the real-world model, so the pilot does not show whether five-token prefixes still collapse under the longer schedule. The simulation controls and the real-world evaluation also differ in data, training budget, environment, and scoring (binary success in simulation versus 1, 0.5, or 0 points per trial on the robots), so their results are not directly comparable, and we ran no matched dense or sparse-global control on the physical robots. Deciding whether the collapse of five-token prefixes is caused by the short training budget would require a matched 10,000-step simulation control on the LIBERO demonstrations, which we leave to future work. Other explanations we cannot exclude are unsuitable hyperparameters (learning rate, gate initialization), an information bottleneck of five tokens that no amount of training would remove, and poor retrieval scores from the mean-embedding context; an inference-time diagnostic that evaluates the trained five-token adapters with all tokens, which we planned but did not complete, would help separate these cases. MVSM is agnostic to the proposal source, but we measured only latency, not task success, for other segmentation backbones (Appendix A.1). Deployment on low-compute edge devices therefore remains challenging. Additionally, our evaluation was restricted to parallel-jaw manipulators; generalization of the sparse gating mechanism to novel robot morphologies, such as

multi-fingered dexterous hands or quadrupedal platforms, remains an open challenge. Finally, we did not evaluate the long-horizon suites, and fine-grained perception alone does not address long-horizon temporal reasoning.

7 Conclusion and Future Work

This work asks whether the dense visual prefix of a pretrained VLA can be replaced by a small set of instruction-retrieved tokens to cope with visual clutter. We propose DFA-VLA, which builds global and segmentation-derived local candidates (Multi-scale Visual-Semantic Modeling, MVSM) and retrieves and gates five of them with instruction-driven scoring (Dynamic Fine-grained Alignment and Fusion, DFAF). In a paired clutter study on LIBERO-Spatial, a LoRA dense control reached 26/32 successes against 22/32 for the published OpenVLA (exact McNemar $p = 0.29$), whereas five-token prefixes collapsed: 3/32 without segmentation and 1/32 for DFA-VLA under a 1,000-step, single-seed budget. In real-world evaluations on 22 tasks, DFA-VLA trained for 10,000 steps (5,000 on LIBERO, then 5,000 on BridgeData V2) obtained a mean score of 78.6%, against 71.1% for the published OpenVLA policy (higher on 16 of 22 tasks), but this result and the simulation pilot differ in budget, environment, and scoring and are not directly comparable. We therefore report the simulation result as a negative pilot finding, together with the measured cost of the pipeline (about 3.3 s per step), and leave a matched 10,000-step simulation control and multiple seeds to future work.

Future work includes evaluating on real-world tasks with higher semantic ambiguity and combinatorial structure, exploring task-level abstractions beyond incremental architectural changes to existing VLAs such as OpenVLA [4], and integrating language models that let robots ask for clarification on ambiguous instructions.

References

[1] OpenAI, "Gpt-4 technical report," 2023.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[3] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," `https://arxiv.org/abs/1910.10683`, 2019, accessed: 2024-05-01.

[4] OpenVLA Authors, "OpenVLA: An Open-Source Vision-Language-Action Model for Robotic Manipulation," 2023, preprint.

[5] RT-2 Authors, "RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control," 2023.

[6] VIMA Authors, "VIMA: General Robot Manipulation with Multimodal Prompts," 2022.

[7] S. M. LaValle, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 17, no. 7, pp. 739–758, 1998.

[8] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[9] Octo Authors, "Octo: A generalist model for robotic manipulation with multi-task learning," 2023, preprint.

[10] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 936–944, 2017.

[11] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8759–8768, 2018.

[12] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 9992–10002, 2021.

[13] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W. Lo, P. Dollár, and R. B. Girshick, "Segment anything," 2023.

[14] M. Oquab, X. Bouthillier, T. Darcet, R. Lenoir, J. Mairal, S. Piantadosi, E. Ripser, M. Szafraniec, J. Verbeek, and F. Bach, "Dinov2: Learning robust visual features without supervision," 2023.

[15] S. Höfer, K. Bekris, A. Handa, J. C. Gamboa, M. Mozifian, and F. Golemo, "Sim2real in robotics and automation: Applications and challenges," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 2, pp. 398–400, 2021.

[16] D. Ho, K. Rao, Z. Xu, E. Jang, M. Khansari, and Y. Bai, "Retinagan: An object-aware approach to sim-to-real transfer," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[17] K. Leung, N. Aréchiga, and M. Pavone, "Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods," 2022, preprint. Published in *International Journal of Robotics Research*, 2022.

[18] H. F. Team, "Parameter-efficient fine-tuning for large models," 2023. [Online]. Available: `https://github.com/huggingface/peft`

[19] Z. Liu, J. Zhang, K. Asadi, Y. Liu, D. Zhao, S. Sabach, and R. Fakoor, "Tail: Task-specific adapters for imitation learning with large pretrained models," 2023.

[20] E. Frantar, S. Ashkboos, T. Hoefler, and D. Alistarh, "Gptq: Accurate post-training quantization for generative pre-trained transformers," 2022.

[21] A. Kulkarni et al., "Awq: Activation-aware weight quantization," 2024.

[22] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi et al., "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[23] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone, "Libero: Benchmarking knowledge transfer for lifelong robot learning," 2023. [Online]. Available: `https://arxiv.org/abs/2306.03310`

[24] C. Mu, Q. Xu, Y. Zhu et al., "Bridgedata v2: A dataset for multi-scene decision making and generalization in robotics," 2023.

[25] Emergen, "Collaborative robot series," `https://www.emergen.cn/products/basic-products/CR/22.html`, 2024, accessed: 2024-07-31.

[26] LEBAI, "LM3 collaborative robot," `https://lebai.ltd/en/`, 2022, accessed: 2024-07-31.

A Additional Results

In total, we evaluate each generalist robot manipulation policy on 22 tasks with 10 trials each, with 6 visual generalization tasks, 4 motion generalization tasks, 4 physical generalization tasks, 4 semantic generalization tasks, and 4 language grounding tasks. As shown in Table 2, all 22 tasks are summarized below, and the values in the results represent the task success rate and standard deviation of the actual test sample:

1. **Put Light Bulb on Brown Plate (Easy Version):** The goal of the robot is to place the light bulb on the brown plate. This task is the Easy Version in visual generalization because the plate used in the test differs in appearance from the one used during training. To simplify the difficulty, the robot's end effector is initially positioned above the light bulb, so the main challenge lies in precise

Table 2: Real-world manipulation performance across 22 tasks. We report per-task scores (out of 10 trials; 1 point per success, 0.5 per partial success) across five generalization categories. Higher scores indicate better performance.

Category	Task	# Trials	RT-1-X	Octo	RT-2-X	OpenVLA	DFA-VLA (10k)
Visual Gen	Put Light Bulb on Brown Plate (Easy)	10	1	5	7	10	**10**
	Put Light Bulb on Brown Plate (Cluttered)	10	0	1	5	9	**9.5**
	Put Teacup from Basket into Sink	10	1	0	0	7	**9**
	Put Apple into Bowl (w/ Clutter)	10	1	3.5	6	6.5	**8.5**
	Put Yellow Bread on Pink Plate	10	1	4	8	**9**	8.5
	Put Yellow Bread on Pink Plate (Shiny)	10	1	3	7	7	**9**
Motion Gen	Lift Green Block	10	3	0.5	6.5	7.5	**9**
	Put Apple on Brown Plate (Height Change)	10	2	1	4.5	4.5	4.5
	Stack Red Cube on Green Cube	10	2	0	8	9	**10**
	Move Spoon from Table to Mug	10	4	1	7	7	**8**
Physical Gen	Put Apple on Brown Plate	10	1	0	1	7.5	**8**
	Flip Teacup Upright	10	2	6	4.5	8	**9**
	Lift Dragon	10	0	1	2.5	7	7
	Open Plastic Jar (Twist)	10	1	1	2	5.5	**6**
Semantic Gen	Move Skull into Drying Rack	10	1	0	5	5	5
	Lift Solid Glue	10	3	0	0	2	**2.5**
	Take Meat Slice out of Brown Plate	10	6	0	5	5	**5.5**
	Stack Green Cup on Yellow Cup	10	0.5	0	5.5	5	**6**
Language Grounding	Put {Red, Green} Block into Basket	10	2.5	4	8.5	7.5	**9**
	Lift {Green Block, Yellow Dinosaur}	10	1.5	2.5	8.5	10	10
	Put {Red, Green} Block on Plate	10	5	5.5	8.5	9.5	**10**
	Bring Either Pen or Pencil	10	2	3	6	8	**9**
Mean Success Rate (%) ± StdErr			18.9±2.6	19.1±2.6	52.7±3.4	71.1±3.1	**78.6±2.8**

Mean success rates and standard errors are computed across all 220 trials of each approach (1 point per success, 0.5 per partial success), following the protocol of OpenVLA [4]; the standard error is approximated as $\sqrt{\hat{p}(1-\hat{p})/220}$. For example, DFA-VLA obtains $173/220 = 78.6\%$ (approximate 95% Wilson interval [72.8%, 83.5%]) and OpenVLA $156.5/220 = 71.1\%$ ([64.8%, 76.7%]); category-level aggregates are in Table 4.

lowering.

2. **Put Light Bulb on Brown Plate (Cluttered):** Same as the Easy Version, but multiple miscellaneous objects are added to the scene as distractions. The robot first needs to identify the light bulb in the complex background, then plan a path to avoid obstacles for placement. Partial score (0.5 points): a partial reward is given when the robot moves toward the correct light bulb but fails to complete the placement.

3. **Put Teacup from Basket into Sink:** The robot needs to grasp a teacup from a basket containing various items and place it into the sink. The challenges lie in correctly identifying the teacup and avoiding other miscellaneous objects, while controlling the placement posture to prevent the teacup from colliding with the edge of the sink.

4. **Put Apple into Bowl (w/Clutter):** The robot's goal is to put the apple into the bowl, and there are several other miscellaneous items piled up in the scene. This task tests the model's capabilities in visual segmentation and grasping planning under a distracting background. Partial score (0.5 points): a partial reward is given when the robot successfully grabs the apple and moves toward the bowl but fails to put it in completely due to path or posture issues.

5. **Put Yellow Bread on Pink Plate:** The robot needs to place the yellow bread on the pink plate. This task evaluates the model's generalization ability in terms of color and shape, especially when the same color combination has not appeared in the training set.

6. **Put Yellow Bread on Pink Plate (Shiny Surface):** Same as the previous task, but due to factors such as lighting, the environment has strong reflective properties. The model needs to overcome visual artifacts, accurately locate the edge of the plate, and complete the placement.

7. **Lift Green Block:** The robot needs to vertically lift the green square off the tabletop. This basic task tests the generalization ability of the action strategy to different positions and postures of objects.

8. **Put Apple on Brown Plate (w/ Height Change):** The robot must first lift the apple to a certain height and then place it on brown plates of different heights. The challenge lies in planning the continuous movement between the lifting height and the placing height.

9. **Stack Red Cube on Green Cube:** The robot needs to accurately place a red cube on top of another green cube to form a stable stack. This tests the model's generalization performance in terms of precise alignment and fine-tuning control.

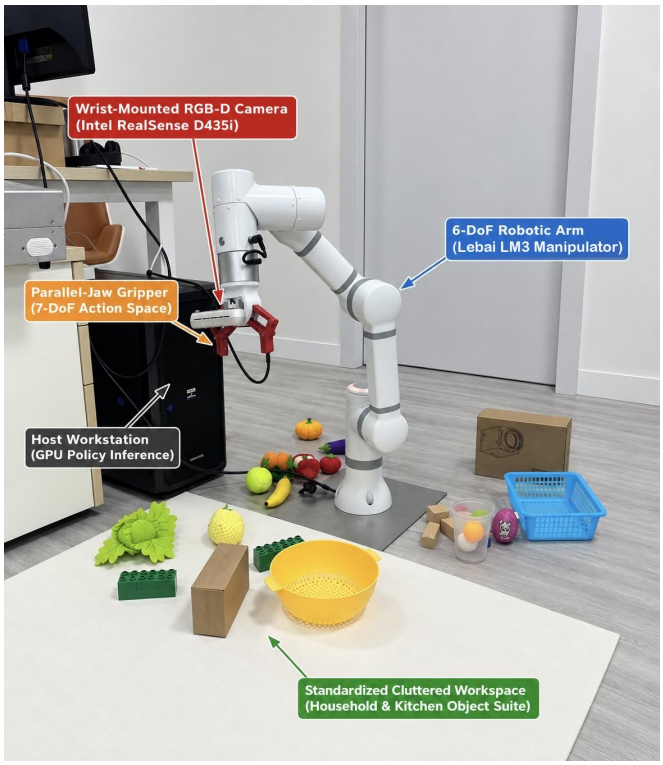


Figure 4: Hardware setup used for physical-robot evaluation featuring Zhichang CR5 and Lebai LM3 manipulators.

10. **Move Spoon from Table to Mug:** The robot needs to move a spoon from the table into a mug. This tests the model's motion generalization performance in terms of precise alignment and fine-tuning control.
11. **Put Apple on Brown Plate:** The robot's goal is to stably place the apple in the center of the brown plate. This task tests the model's physical generalization ability in terms of contact dynamics (gravity, friction).
12. **Flip Teacup Upright:** In the initial state, the teacup is upside down. The robot needs to first grasp the teacup and flip it to a normal upright state before placing it on the table. The key points lie in the selection of the grasping point and the posture control during the flipping process.
13. **Lift Dragon:** The robot must lift the irregularly shaped dragon toy. This task tests the model's ability to handle contact dynamics of non-rigid or complex geometric objects.
14. **Open Plastic Jar (Twist Motion):** The robot needs to first grasp the cap of the plastic jar, then open it through a rotational movement and put the cap back. This task emphasizes torque control and the stability of continuous rotational actions.
15. **Move Skull into Drying Rack:** The robot needs to identify the skull model and place it into the designated grid of the drying rack. This task tests how the model maps the semantic concept to the correct object and completes spatial positioning.
16. **Lift Solid Glue:** The robot needs to identify the solid glue stick and lift it to the work area. This task verifies the model's semantic recognition of objects with different materials and the generalization of grasping strategies.
17. **Take Meat Slice out of Brown Plate:** The robot needs to identify the meat slices in the brown plate, take them out, and place them in another container. This poses challenges to the model's semantic segmentation and safe grasping.
18. **Stack Green Cup on Yellow Cup:** The robot first needs to identify two semantically distinct cups (green and yellow), then stack the green cup on top of the yellow cup. This tests the model's generalization ability in hierarchical semantic operations. Partial score (0.5 points): a partial reward is given when the robot correctly identifies and grabs the green cup but fails to complete the stacking placement.

19. **Put {Red Block, Green Block} into Basket:** Based on the semantic variables in the instruction, the robot needs to put the red or green square into the basket. This task evaluates the model's ability to parse and execute multiple options in parentheses within the language.
20. **Lift {Green Block, Yellow Dinosaur}:** The robot lifts the specified target according to natural language instructions: either the green square or the yellow dinosaur toy. This tests the model's accuracy in parsing multiple-choice semantics and mapping actions.
21. **Put {Red Block, Green Block} on Plate:** Similar to Task 19, but the placement target is changed to a plate. The robot needs to place the specified square in the center of the plate according to the instruction and avoid putting the wrong square onto it.
22. **Bring Either Pen or Pencil:** The robot needs to follow the instruction and bring back either a pen or a pencil. This task tests the model's language comprehension and object retrieval abilities in either/or scenarios. Partial score (0.5 points): a partial reward is given when the robot finds the correct option but fails to bring it back due to path planning or grasping failures.

A.1 Segmentation Backbone Latency

MVSM is agnostic to the proposal backbone. Table 3 reports latencies that we measured at 224×224 input in float32 (PyTorch 2.2.0, CUDA 12.1), using automatic mask generation with a 16×16 point grid and 64 points per batch. Values are medians (95th percentile in parentheses) over 50 (MobileSAM, A100), 30 (MobileSAM, A40), and 10 (SAM ViT-H, A40) repeated runs; the complete pipeline (MobileSAM masks, crop re-encoding, and the OpenVLA policy) was measured over 10 runs on the A100. We did not evaluate task success for different segmentation backbones and did not measure FastSAM.

Table 3: Measured segmentation and pipeline latency. FPS is the reciprocal of the total latency.

Backbone	**GPU**	**Encoder (ms)**	**Mask generation (ms)**	**FPS**
MobileSAM	A100	22.9	300.8 (306.2)	3.3
MobileSAM	A40	27.7	433.6 (437.7)	2.3
SAM ViT-H	A40	487.0	827.1 (828.9)	1.2
Complete DFA-VLA pipeline	A100	—	3305.1 (3447.0)	0.30

A.2 Statistical Methodology: Wilson 95% Score Intervals

For s successes in n trials, let $\hat{p} = s/n$ and $z = 1.96$. The Wilson score interval is

$$\mathrm{CI} = \frac{\hat{p} + z^2/(2n) \;\pm\; z\sqrt{\hat{p}(1-\hat{p})/n + z^2/(4n^2)}}{1 + z^2/n}. \tag{4}$$

Because the real-world protocol awards 0.5 partial credit, its success counts can be non-integer and the binomial assumption is approximate. We therefore report these intervals as approximate.

Table 4: Real-world category breakdown with approximate 95% Wilson intervals (Appendix A.2).

Category	**Tasks**	n	**SR**	**95% CI**
Visual Gen.	6	60	90.8%	[80.9, 95.9]
Motion Gen.	4	40	78.8%	[63.9, 88.6]
Physical Gen.	4	40	75.0%	[59.8, 85.8]
Semantic Gen.	4	40	47.5%	[32.9, 62.5]
Language Grounding	4	40	95.0%	[83.5, 98.6]
Overall	22	220	78.6%	[72.8, 83.5]

Per-task intervals at $n = 10$ are wide, so conclusions are drawn at category and overall levels rather than from individual tasks.

A.3 Real-World Failure Mode Taxonomy

Based on trial-video review, we attribute roughly 47 failure or partial-failure episodes (partial-credit trials are pooled with the failures; DFA-VLA lost 47 of 220 points in total) to three primary physical modes. The counts are approximate. *Contact slip* denotes loss of a correctly selected object during grasp or transport. *Occlusion drift* denotes target localization drifting to an occluder or background region. *Depth misestimation* denotes incorrect vertical approach, contact, or release height.

Table 5: Approximate real-world failure-mode counts across roughly 47 failure or partial-failure episodes of DFA-VLA, from trial-video review; partial-credit trials are pooled with failures.

Mode	**Operational definition**	**Approx. count (%)**
Contact slip	Object lost during grasp, transport, or release	21 (44.7%)
Occlusion drift	Target localization shifts under clutter or occlusion	16 (34.0%)
Depth misestimation	Incorrect approach, contact, or release height	10 (21.3%)

B Datasets

To study the proposed method in robotic manipulation tasks, this paper conducts experiments on several representative benchmark datasets, including the OXE dataset and the LIBERO dataset. These data are mainly used for simulation and real-machine testing of robots in typical living scenarios such as homes and kitchens, covering manipulation tasks of common objects like furniture, food, and tableware. With diverse task settings and sample distributions, they can comprehensively evaluate the generalization ability and transfer effect of the model across tasks and robotic platforms. This section elaborates on the basic information and data characteristics of each dataset, as well as the preprocessing methods adopted in the experiments.

B.1 Open X-Embodiment (OXE) Dataset

The Open X-embodiment (OXE) dataset is a large-scale, multi-robot platform robotic manipulation dataset, aiming to advance research on cross-robot general strategies. It is collected by 22 different robots from 21 institutions, covering 527 skills and over 160,000 task instances, and uniformly adopts a standardized data format, supporting modeling for a wide range of manipulation tasks.

In the real-world experiments, the model was trained for 5,000 steps on LIBERO followed by 5,000 steps on BridgeData V2, an important subset of the OXE dataset; no self-collected data was used. This subset focuses on robots' manipulation behaviors in home and kitchen environments, involving a variety of object types such as tableware, food, and furniture, which are close to practical application scenarios. The task composition and data quality of BridgeData V2 are highly suitable for evaluating the generalization ability and transfer performance of the proposed method in diverse environments.

B.2 LIBERO Dataset

LIBERO is a robotic manipulation dataset designed specifically for research on knowledge transfer in multi-task and lifelong learning scenarios. This dataset emphasizes the transferability of two types of knowledge: declarative knowledge about objects and spatial relationships, and procedural knowledge about actions and behaviors. LIBERO provides a programmatic generation framework capable of generating an infinite number of manipulation tasks, and includes 130 tasks divided into four task suites: LIBERO-Spatial, LIBERO-Object, LIBERO-Goal, and LIBERO-100. The first three suites feature controllable distribution shifts, which are used to evaluate the transferability of specific types of knowledge. LIBERO-100, on the other hand, contains 100 tasks to test the generalization ability of models under the cross-integration of knowledge, among which LIBERO-90 is used for pre-training and LIBERO-10 for evaluating the performance of downstream lifelong learning.

We focus on the LIBERO-Spatial suite from the LIBERO dataset for the clutter study. This suite centers on understanding and manipulating object spatial relationships, with spatial position constraints

as its core (e.g., object placement, distance adjustment). Its controllable, spatial layout-based distribution shifts make it ideal for evaluating a model's spatial cognition and generalization, aligning with our method's cross-scene spatial adaptability goal. Unlike other sub-datasets, it uses object spatial layouts and relationships as the core constraint, with single-dimensional, quantifiable shifts, enabling independent assessment of adaptability to pure spatial changes, especially for grasping and placing tasks. We chose it for simulation because: (1) it programmatically generates gradient spatial scenarios, avoiding real-world standardization issues; (2) its precise spatial data is compatible with simulation engines, aiding rapid reproduction and large-scale iterations; and (3) task complexity matches home scenario spatial logic, facilitating strategy transfer to real operations and supporting physical testing.

Ethical Statement

This research complies with responsible research conduct. All experiments were conducted using real and available datasets, standard simulation and physical environments, without involving any sensitive data. No human subjects or animals were involved in the data collection or experimentation.

Our work aims to advance multimodal understanding and robot action planning from natural language instructions, which has potential positive societal impact in areas such as assistive robotics, accessibility technologies, and human-robot interaction.

However, we also recognize the potential negative implications of deploying such systems without proper safeguards. Risks include misuse in surveillance, physical harm from incorrect robotic execution, or biased behavior inherited from pre-trained models. Specifically, experiments should be conducted under the premise that operators are sufficiently familiar with robotic movement and end-effector operations, ensuring the personal safety of all personnel involved and preventing damage to experimental equipment. To mitigate these risks, we advocate for rigorous downstream safety evaluations, transparency in deployment, and ongoing monitoring of real-world impacts.

AI Use Disclosure

In accordance with ICLR 2027 policy, the authors disclose the use of large language model (LLM) assistance for grammar correction, language polishing, and improving prose clarity. All scientific contributions, including experimental design, data collection, quantitative results, model architecture decisions, and core research hypotheses, are entirely the work of the authors. AI tools were used solely as writing and editing aids and did not generate or verify any experimental results reported in this work.