Specialist VLA: Planner-Routed LoRA Specialization for Long-Horizon Robotic Manipulation

Christian I. Narcia-Macias [1] **Daniel Masamba** [1] **Erik Enriquez** [1] **Dongchul Kim** [1]

Abstract

Long-horizon robotic manipulation requires chaining semantically distinct primitives, reaching, grasping, moving, and placing, where per-primitive failures compound into end-to-end task breakdown. Existing hierarchical approaches use Vision-Language Models (VLM) planners to decompose instructions but route all primitives through a single generalist low-level controller, creating a planner-executor mismatch that limits reliability. We propose a Specialist Vision Language Action Model (VLA), which retains a shared frozen TinyVLA-1.3B backbone while activating primitive-specific LoRA adapters selected by a Gemini-based planner, with dynamic re-querying every 30 control steps for mid-execution recovery. On a pick-and-place benchmark in Robosuite, Specialist VLA achieves 90% full-task success with re-querying versus 70% for a generalist baseline under identical re-querying conditions, demonstrating that adaptive planning yields substantial gains only when paired with primitive-level specialization.

1. Introduction

Recent progress in robot learning has been driven by the emergence of vision language action (VLA) models, which integrate large scale semantic knowledge from vision language pretraining with low level action generation. This paradigm marks a shift away from hand-engineered, task specific robotics pipelines toward generalist foundation models that can interpret natural language instructions and act directly from visual observations. Models such as RT-2, OpenVLA, π_0, and related systems suggest that a single pre-trained policy can support broad manipulation competence across tasks, embodiments, and interaction settings. In this paper, we do not argue against generalist VLAs, rather, we take them as the right foundation ask how their control architecture should evolve as manipulation tasks become longer horizon, more compositional, and more precision-sensitive.

Despite this progress, long horizon manipulation remains brittle. Many real world tasks require chaining together semantically distinct subtasks such as picking, opening, pouring, inserting, or placing, and success depends on executing each stage reliably under changing scene conditions. In these settings, small local execution errors do not remain local, they propagate into later stages, degrade state quality, and eventually collapse end-to-end performance (Han et al., 2025).

This is investigated through a specialist VLA with a hierarchical architecture in which a VLM acts as a planner that decomposes an instruction into primitive labels, and a lightweight VLA backbone uses those labels to route execution through primitive-specific LoRA adapters. This design is intended to improve primitive reliability without sacrificing the scalability and generalization benefits that make VLAs attractive in the first place.

This paper is organized around three questions. First, do generalist VLAs exhibit systematic performance gaps across primitive tasks? Second, can primitive typed attention head specialization reduce these gaps while using a lightweight backbone? Third, do primitive level gains compound into meaningfully better long horizon task completion? To answer these questions, we evaluate specialist VLA using decomposed primitive level analysis together with long-horizon benchmarks, with the goal of showing that specialization at the controller level addresses a bottleneck that is complementary to advances in planning, reasoning, and recovery.

2. Background

In language conditioned robotic manipulation, a policy receives visual observations o_t, an instruction x, and optionally additional state inputs such as proprioception or end-effector pose, and predicts an action a_t at each timestep. A

[1]Department of Computer Science, The University of Texas Rio Grande Valley, Edinburg, TX, USA. Correspondence to: Christian I. Narcia-Macias <christian.narcia01@utrgv.edu>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

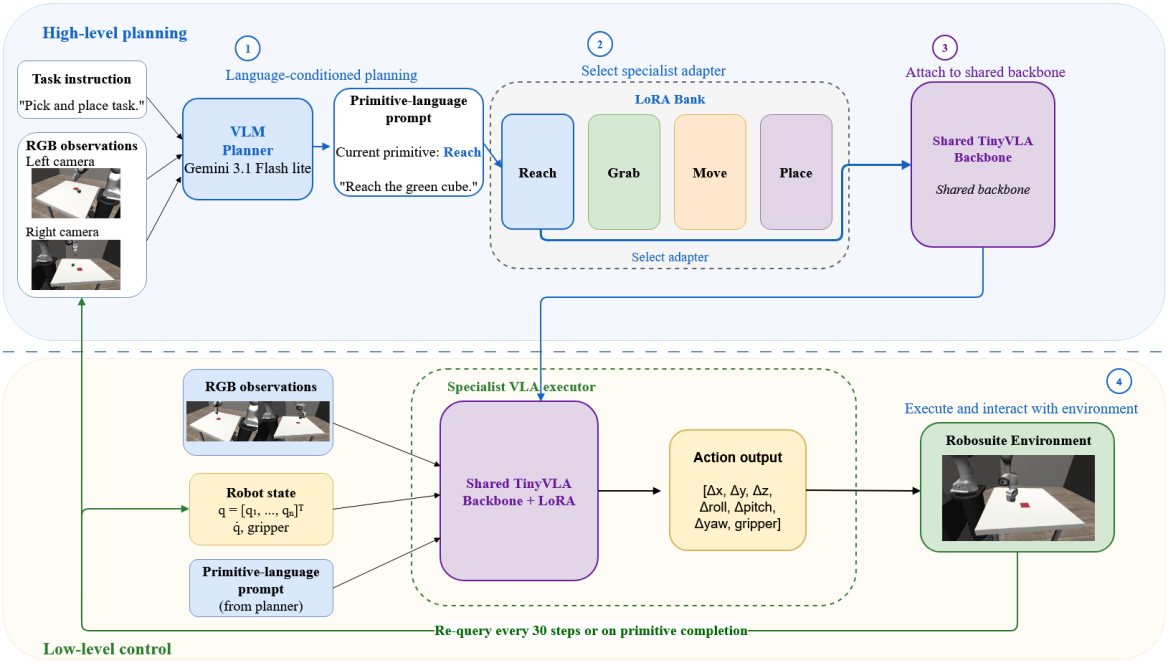

Figure 1. A VLM planner decomposes a natural language instruction into motion primitives**(1–2)**, selecting a specialist LoRA adapter from a frozen bank (Reach, Grab, Move, Place) and composing it with a shared frozen TinyVLA backbone **(3)**. The resulting specialist executor takes multi-view RGB observations, proprioceptive robot state, and a primitive-language prompt to output end-effector delta actions**(4)**

trajectory or episode is therefore represented as a sequence of observations, actions, and task context over time. In the long-horizon setting, the task cannot be treated as a single atomic behavior, but is instead naturally viewed as a composition of multiple subtasks that must be completed in sequence.

A VLA model maps visual observations and language instructions directly to robot actions, allowing semantic task understanding and low level control to be learned within a shared policy. Recent VLA systems combine pretrained vision language backbones with action generation modules, making it possible to execute manipulation tasks from natural language. In most existing VLA architectures, however, the low level controller remains largely shared across tasks and subtasks, even when the model is evaluated in diverse multi task or long horizon settings.

This method builds on the generalist VLA paradigm rather than replacing it: the core foundation-model backbone remains shared, while primitive-specific LoRA adapters provide lightweight specialization during execution.

The term primitive refers to a low-level manipulation type, such as `reach`, `place`, `grab`, `pour`, or `open`. In this framework, primitives are more structured than raw motor commands. A primitive identifies the current manipulation mode that the robot should execute at a given stage of the task. By decomposing a long-horizon instruction into a sequence of primitives, the system creates a compact execution plan that determines which specialized adaptation module should be used during each stage.

The methodology is motivated by the observation that different manipulation primitives exhibit different action distributions, contact dynamics, visual cues, and failure modes. A single generalist controller must represent all of these behaviors through the same adaptation pathway, which can lead to cross-primitive interference and uneven reliability. Instead of duplicating the full VLA controller or training separate full models for each primitive, this approach retains a shared frozen backbone and trains lightweight LoRA adapters specialized for individual primitive types.

During execution, the high-level planner predicts the currently active primitive, and the corresponding LoRA adapter is activated within the shared VLA backbone. This allows the policy to use primitive-specific parameter updates while preserving the representational capacity and transfer benefits of the pretrained generalist model. Compared with larger mixture-of-experts designs, planner-routed LoRA specialization is parameter-efficient because only small low-rank adapter modules are specialized, while the majority of the model remains shared.

3. Related Works

3.1. Generalist Vision Language Action Models

The introduction of large-scale transformer policies trained on diverse robot demonstrations established the generalist-VLA paradigm. RT-1 (Brohan et al., 2022) a large-scale multi-task robotic transformer demonstrated that a single tokenized image based policy could be trained on 700+ manipulation tasks and generalize to novel instructions when scaled appropriately, and RT-2 (Brohan et al., 2023) formalized VLA models extended this by co-fine-tuning a web-pretrained VLM (up to 55B parameters) to emit robot actions as text tokens, transferring emergent semantic knowledge to manipulation. Subsequent work scaled this recipe down without sacrificing capability: OpenVLA (Kim et al., 2024) matched RT-2-X at 7× fewer parameters using an open backbone, while Octo (Octo Model Team et al., 2024) introduced block-wise attention masking over task, observation, and readout tokens. More recent models push toward real world dexterity and generalization, π_0 (Black et al., 2024) couples a PaLI-Gemma backbone with a flow matching action expert, and $\pi_{0\cdot5}$ (Black et al., 2025) extends this to unseen homes. Across all of these systems, however, a single unified policy interface handles every manipulation primitive picking, placing, pouring, opening identically. Efficiency-focused variants such as TinyVLA (Wen et al., 2025) and SmolVLA(Shukor et al., 2025) compress this architecture further, but preserve the same one-size-fits-all action interface.

3.2. Hierarchical Planning and Task Decomposition

The most closely related prior work to a specialist VLA is a family of hierarchical VLA frameworks that introduce hierarchical decompositions in which high-level reasoning modules guide lower-level action policies, Hi Robot (Shi et al., 2025), HiRT (Zhang et al., 2024), and RT-H (Belkhale et al., 2024). These systems correctly identify that a monolithic end-to-end policy often struggle to reliably plan and execute long-horizon tasks simultaneously, and decompose the problem into a slow reasoning loop and a fast execution loop. The planning side has also evolved toward more structured, typed interfaces. Code as Policies (Liang et al., 2023) and ProgPrompt (Singh et al., 2023) express subtasks as typed function calls, while VoxPoser (Huang et al., 2023) generates stage specific constraint maps. In other words, the system may reason in typed subtasks, such as distinguishing between precision grasping, compliant pouring, or articulated door opening, but execution is typically implemented through a shared controller that handles all primitive types using the same underlying policy, with all primitives routed through the same shared controller. This can creates a planner executor mismatch, where the planner explicitly represents differences between primitive types, yet the executor lacks specialized mechanisms to act on those differences. We argue that this mismatch is a key limitation: typed planning can only improve long-horizon reliability if the executor preserves primitive distinctions and executes each type using behaviors appropriate to its physical demands.

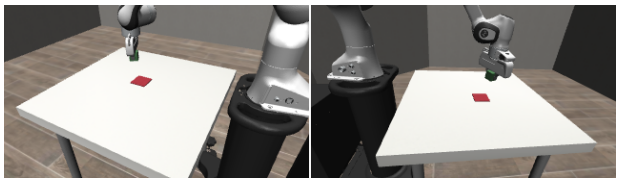

Figure 2. Left and Right Camera Image from Simulation

3.3. Long-Horizon Manipulation and Compounding Error

The compounding of local errors over long task horizons is a central failure mode in imitation learning. Ross et al. (Ross et al., 2011) showed that standard behavioral cloning can suffer horizon dependent error amplification: because the learner's actions affect future observations, even small deviations from the expert distribution can induce covariate shift and lead to $O(T^2\epsilon)$ style performance degradation over horizon T. This theoretical issue appears directly in long horizon robot benchmarks. CALVIN (Mees et al., 2022) evaluates agents on 1,000 chains of five language conditioned tasks, where success is reported as the number of consecutive subtasks completed; even strong policies exhibit a gap between single-step and five-step completion. LIBERO-Long (Liu et al., 2023) similarly stresses policies through 10 composite household manipulation tasks. More recent benchmarks such as RoboCerebra (Han et al., 2025) further increase the temporal and compositional burden: its tasks contain 2 to over 20 atomic steps, average 2,972.4 simulation steps, span 12 primitive action types, and over 10% of tasks require five or more distinct action categories. These settings suggest that long horizon success is not determined only by high level planning, but also by the reliability of each primitive controller. Under finite robot data and heterogeneous primitive distributions, improving per-primitive reliability becomes a direct lever for improving end-to-end task success.

4. Methods

4.1. Problem Formulation

This work studies long-horizon robotic manipulation tasks that consist of sequences of semantically distinct primitive actions. Let o_t denote the observation at timestep t, consisting of two RGB image observations, a language instruction describing the current task, and proprioceptive robot

state, including end-effector pose and gripper state. A long-horizon task can be expressed as an ordered sequence of primitives,

$$z_1, z_2, \ldots, z_K,$$

where each primitive $z_i \in \mathcal{Z}$ corresponds to a temporally extended manipulation skill with its own control distribution and failure modes. In this work, the primitive set is instantiated as

$$\mathcal{Z} = \{\texttt{reach}, \texttt{grab}, \texttt{move}, \texttt{place}\},$$

corresponding to the four motor phases of a pick-and-place task.

Existing VLA policies often employ a single shared controller to execute all primitives, despite the fact that each primitive z_i exhibits distinct low-level action statistics and failure modes. This undifferentiated control structure can introduce cross-primitive interference and uneven primitive-level reliability, which may compound over long task horizons. The objective is to learn a hierarchical policy that first decomposes a high-level instruction into a primitive sequence using a language planner, then routes execution through primitive-specific adaptation pathways while preserving parameter efficiency through a shared frozen backbone.

4.2. Framework Overview

Specialist VLA consists of two components operating at different temporal scales: a high-level planner that segments the task into primitives, and a low-level controller that executes each primitive through a dedicated LoRA adapter. At inference time, the planner identifies the current primitive z_t and activates the corresponding adapter. The low-level policy then executes under that adapter until a completion condition is satisfied or a fixed step budget is exceeded, at which point the planner is re-queried.

4.3. High-Level Planner

The Gemini API (Google DeepMind, 2025) is used as the high-level planner. Given a natural-language task description and the current visual observation, the planner decomposes the overall task into a sequence of primitives drawn from $\mathcal{Z}$ and identifies the currently active primitive z_t.

The planner is queried under two conditions: (1) every $T = 30$ control steps, and (2) when primitive completion is detected through scripted environment conditions. Periodic re-querying provides a recovery mechanism: if primitive execution fails or produces an unexpected state, the planner can revise the active primitive prediction at the next query rather than continuing with a stale plan. This distinguishes the proposed approach from static task-planning methods that fix the primitive sequence at the beginning of execution and do not update the plan in response to low-level execution failures.

4.4. Shared VLA Backbone

The lightweight policy backbone used in this project is based on TinyVLA-1.3B (Wen et al., 2025). This model was selected because it provides a practical balance between model capacity and computational efficiency, making it suitable for parameter-efficient specialization with primitive-specific LoRA adapters. Compared with larger VLA backbones, TinyVLA-1.3B is easier to fine-tune, less expensive to evaluate across multiple primitive specialists, and well aligned with the goal of testing whether lightweight modular adaptation can improve long-horizon manipulation without requiring a separate large-scale policy for each primitive.

The policy receives multimodal inputs consisting of two RGB image observations, a language instruction describing the current task or primitive, and proprioceptive robot state such as end-effector pose and gripper state. The policy outputs continuous robot control commands in a 7-dimensional action space. At each timestep t, the action is represented as

$$a_t = \begin{bmatrix} \Delta x & \Delta y & \Delta z & \Delta r_x & \Delta r_y & \Delta r_z & g \end{bmatrix},$$

where $\Delta x, \Delta y$, and Δz denote translational end-effector displacement, $\Delta r_x, \Delta r_y$, and Δr_z denote rotational displacement, and g denotes the gripper command. When action chunking is used, the model predicts a short sequence of future actions,

$$\hat{A}_t = \{\hat{a}_t, \hat{a}_{t+1}, \ldots, \hat{a}_{t+H-1}\},$$

rather than a single next-step command, where H is the action chunk horizon.

4.5. Primitive Specific LoRA Adapters

For each primitive $z_i \in \mathcal{Z}$, a dedicated low-rank adaptation (LoRA) module is trained (Hu et al., 2022). Each LoRA module introduces trainable low-rank updates into the frozen VLA backbone according to the standard LoRA formulation. The LoRA rank and scaling hyperparameters are kept consistent with the original TinyVLA configuration (Wen et al., 2025). Following the TinyVLA fine-tuning protocol, the trainable parameters also include a newly introduced policy decoder connected to the backbone by a linear projection layer.

During inference, the planner prediction z_t is used to select the active primitive-specific adapter. Because only one

adapter is active at each timestep, the inference cost remains equivalent to a single-adapter forward pass and does not scale with the total number of available specialists.

4.6. Training

Each LoRA adapter is trained independently using a primitive-specific dataset of scripted demonstrations collected in simulation. The training data for each adapter contains only demonstrations of the corresponding primitive in isolation. For example, the `grab` adapter is trained exclusively on trajectories from the grasping phase, without exposure to `reach`, `move`, or `place` transitions. This isolates the behavioral distribution that each adapter must capture and reduces cross-primitive interference compared with a shared generalist policy.

Formally, for each primitive $z_i \in \mathcal{Z}$, let

$$\mathcal{D}_{z_i} = \{(o_t, a_t)\}$$

denote the dataset of observation-action pairs associated with primitive z_i. The adapter for primitive z_i is trained using the behavioral cloning objective

$$\mathcal{L}_{z_i} = \mathbb{E}_{(o_t,a_t)\sim\mathcal{D}_{z_i}} \left[\|\pi_\theta(o_t; \phi_{z_i}) - a_t\|_2^2 \right],$$

where θ denotes the frozen backbone parameters and ϕ_{z_i} denotes the trainable LoRA and decoder parameters associated with primitive z_i.

4.7. Experimental Setup

Environment Experiments are conducted in Robosuite (Zhu et al., 2020) using the MuJoCo physics engine (Todorov et al., 2012). The current benchmark consists of a pick-and-place task in which a robot arm must pick up a cube and place it onto a target area. Task success is evaluated using scripted environment conditions that compare the final object pose against predefined goal thresholds.

Evaluation Protocol The primary evaluation metric is full-task success rate, measured across N evaluation rollouts. A rollout is considered successful only if all primitives in the sequence are completed correctly and the final task goal is achieved.

Baselines The proposed method is compared against the following baselines:

- **Generalist TinyVLA:** A single TinyVLA model fine-tuned on demonstrations from all primitives combined, without primitive-specific specialization.
- **OpenVLA-OFT:** A full-scale VLA baseline fine-tuned on the complete pick-and-place task, representing a stronger single-model baseline.
- **Specialist TinyVLA without planner re-querying:** An ablation in which the primitive sequence is fixed at the beginning of execution. This baseline isolates the contribution of dynamic planner re-querying from the contribution of primitive-specific LoRA specialization.

For Generalist TinyVLA, 1,000 demonstrations per primitive are combined into a single dataset of 4,000 episodes total, trained under a unified policy. For specialist TinyVLA, each primitive-specific LoRA adapter is trained independently on 1,000 demonstrations of its corresponding primitive in isolation, matching the per-primitive data volume of the generalist baseline. OpenVLA-OFT is trained on 1,000 complete pick-and-place episodes. This comparison is fair despite the difference in total episode count: primitive-level demonstrations are significantly shorter in step length than full task episodes, making the total environment interaction comparable across conditions. The 1,000 episode budget for OpenVLA-OFT reflects the practical constraint that full task demonstrations are expensive to collect, and intentionally evaluates all models in a low-data regime relevant to real-world deployment.

5. Results

Specialist VLA is evaluated against Generalist TinyVLA, which uses the same frozen backbone and Gemini planner but relies on a single unified controller for all primitives. We test both controllers under two planning conditions: a fixed primitive sequence and dynamic re-querying during execution. This lets us compare the effect of controller specialization while also measuring how much dynamic planner feedback improves performance. The fixed-sequence Specialist VLA condition serves as an ablation that isolates the contribution of re-querying. We also include OpenVLA-OFT, a larger monolithic VLA fine-tuned on full pick-and-place demonstrations in the same low-data setting; because it differs from the TinyVLA conditions in parameter scale, architecture, and demonstration format, it serves as a data-efficiency reference rather than a controlled baseline. Full-task success rate is the primary metric, measured across N rollouts with randomized cube and target positions, and we additionally analyze per-primitive reliability to identify where specialization most improves execution quality.

Table 1 breaks down success rate at the primitive level for Generalist TinyVLA and Specialist TinyVLA. While full-task success rate captures overall performance, per-primitive analysis reveals where failures originate and how they propagate. A generalist policy trained across all primitives jointly is expected to exhibit uneven reliability across manipulation

Table 1. Per-primitive success rate (%) for Generalist TinyVLA and Specialist TinyVLA (ours). Each primitive is evaluated independently across all rollouts.

Primitive	Gen TinyVLA	Spec TinyVLA (ours)
Reach	94	**95**
Grab	85	**97**
Move	86	**92**
Place	89	**90**
Average	88.5	**93.5**

phases, with higher failure rates concentrated in contact-rich primitives such as `grab` and `move` where precise low-level control is most critical. Specialist TinyVLA, by contrast, trains each adapter on an isolated primitive distribution, which we hypothesize reduces interference from dissimilar action patterns and improves reliability at the phases most prone to failure. Because long-horizon success requires every primitive in the sequence to complete correctly, even modest per-primitive improvements compound into larger gains at the task level, a failure at `grab`, for example, renders all subsequent primitives unreachable regardless of their individual reliability.

Table 2. Full task success rate (%) across all models over 100 evaluation rollouts with randomized cube and target positions. Generalist TinyVLA and Specialist TinyVLA (ours) both fixed primitive sequence at inference and does not update based on environment state; they differ only in whether the low-level controller is shared or primitive-specific.

Model	Full Task SR (%)	Model Param
Gen TinyVLA (fixed)	62	1.3B
Spec TinyVLA (fixed)	**78**	**1.3B**

Table 2 reports full-task success rates across all models over N evaluation rollouts with randomized cube and target positions. Specialist TinyVLA outperforms Generalist TinyVLA despite using the same frozen backbone, planner, parameter scale, and training data volume. This result suggests that primitive-specific LoRA specialization improves long-horizon reliability beyond what is achieved by a single generalist adaptation pathway.

The comparison between specialist TinyVLA and Generalist TinyVLA is the most controlled evaluation, since the backbone weights, Gemini-based routing mechanism, and demonstration budget are held constant. The primary difference is the use of primitive-specific LoRA adapters rather than one shared policy adaptation pathway. Therefore, the performance gap isolates the contribution of primitive-level specialization.

For additional context, we evaluate against OpenVLA-OFT, a 7B parameter model trained on 1,000 full-task demonstrations, intentionally conducted in a low-data regime reflecting the practical constraint that robot demonstrations are expensive to collect at scale. OpenVLA-OFT is a stronger architecture in principle, but its fine-tuning recipe is designed for larger demonstration sets; under 1,000 episodes it fails to reach competent performance. This is not presented as evidence of architectural inferiority, but rather as a data efficiency comparison: primitive-specific LoRA specialization over a lightweight backbone achieves reliable task completion at a demonstration budget where a larger monolithic fine-tuning approach does not. This property is particularly relevant for real-world deployment scenarios where per-task demonstration collection is a practical bottleneck.

Table 3. Full task success rate (%) with dynamic planner re-querying. With re-querying, the planner is polled after each primitive execution and adapts accordingly, subject to a 10-attempt budget per episode. Cube and target positions are randomized at the start of each episode.

Condition	Full Task SR (%)
Generalist TinyVLA (re-querying)	70
Specialist TinyVLA (re-querying)	**90**

Table 3 reveals two complementary findings. First, re-querying alone provides minimal benefit to the generalist controller, improving full-task success by 8 percent. Second, when combined with primitive-specific LoRA specialization, re-querying yields a substantial gain, bringing Specialist TinyVLA from 78% to 90%. This asymmetry suggests that re-querying's value depends on the quality of the underlying executor: if the low-level controller cannot reliably execute a correctly identified primitive, mid-execution plan revision offers little recovery. Dynamic re-querying partially mitigates execution drift by allowing the system to reassess the current scene and select the primitive that best matches the observed state, rather than blindly advancing through a stale sequence. Together, these results indicate that primitive-level specialization and adaptive planning are complementary bottlenecks rather than interchangeable improvements.

6. Discussion and Limitations

The results support the central hypothesis that long-horizon manipulation performance depends strongly on primitive-level reliability. Planner-routed LoRA specialization improves full-task success by addressing failures at the subtask level rather than relying on a single generalist policy to handle all manipulation modes. Critically, this specialization is introduced without training separate full models per primitive, the shared frozen backbone preserves the representational capacity of the pretrained VLA, while the LoRA adapters provide lightweight primitive-specific adjustments. This positions specialist VLA between fully generalist and

fully modular approaches: more structured than a single fine-tuned policy, but more parameter-efficient than a suite of independent specialists.

Dynamic planner re-querying contributes an important recovery capability. By updating the active primitive every 30 steps rather than committing to a fixed plan at execution onset, the system can partially recover from low-level deviations that would otherwise propagate irreversibly across the task horizon. This suggests that the boundary between planning and control in hierarchical systems may benefit from being more fluid than is typically assumed. More broadly, these findings indicate that hierarchical planning alone is insufficient for reliable long-horizon manipulation if all primitives are still executed by an undifferentiated controller. The quality of high-level decomposition and the quality of low-level execution are complementary bottlenecks; improving only one leaves the other as a ceiling on overall performance.

Limitations The current evaluation relies on scripted simulator conditions for primitive completion detection, which does not transfer to real-robot settings where ground-truth object state is unavailable. Evaluation is also scoped to a single pick-and-place task, leaving the generality of primitive specialization across more diverse long-horizon behaviors unvalidated. We do not compare against a large monolithic VLA trained with a demonstration budget matched to its fine-tuning recipe. OpenVLA-OFT is deliberately trained in the same low-data regime as the other models, so it reflects data efficiency at a fixed budget rather than the peak performance a high-capacity VLA could reach with adequate data. Whether primitive-specific specialization retains its advantage against a fully resourced large baseline is therefore left open. Finally, Gemini API latency imposes a practical constraint on re-query frequency that would need to be addressed before real-time deployment.

Future work includes expanding the primitive set to cover articulated object interaction and contact-rich assembly, replacing scripted completion detection with a learned primitive classifier, and incorporating memory-augmented planning to support tasks that require repeated primitive execution or context-dependent sequencing. Evaluating the framework across longer task horizons with larger primitive vocabularies remains an important open question.

7. Conclusion

We presented a specialist VLA, a hierarchical framework for long-horizon robotic manipulation that addresses a fundamental limitation of existing VLA approaches: the use of a single undifferentiated controller across semantically distinct primitive actions. This method retains a shared frozen TinyVLA-1.3B backbone while routing execution through primitive-specific LoRA adapters activated by a Gemini-based high-level planner. By training each adapter in isolation on primitive-specific demonstration data, specialist VLA reduces cross-primitive interference and improves per-primitive execution reliability without the parameter cost of training separate full models.

Experiments on a pick-and-place task in Robosuite demonstrate that specialist VLA achieves higher full-task success rates than a generalist TinyVLA baseline trained on all primitives jointly, and achieves reliable completion with a substantially smaller trainable parameter footprint, and in a data-limited setting where OpenVLA-OFT does not. Per-primitive analysis confirms that the gains are distributed across the manipulation sequence rather than concentrated in a single phase, supporting the hypothesis that specialization reduces compounding errors over the task horizon. The planner's dynamic re-querying further enables mid-execution recovery, improving robustness to low-level failures that would otherwise cause irreversible task breakdown under a fixed plan.

Acknowledgements

This material is based upon work supported by the National Science Foundation (NSF) under Grant No. 2434728. The views and conclusions expressed are those of the authors and do not necessarily reflect the views of the NSF.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning, specifically in the context of long-horizon robotic manipulation. The proposed method improves the reliability of robot control policies through parameter-efficient specialization, with potential applications in automation, assistive robotics, and manufacturing. We do not anticipate immediate negative societal consequences specific to this work beyond those generally associated with advances in robotics and automation. All experiments are conducted in simulation, and no human subjects or sensitive data are involved.

References

Belkhale, S., Ding, T., Xiao, T., Sermanet, P., Vuong, Q., Tompson, J., Chebotar, Y., Dwibedi, D., and Sadigh, D. RT-H: Action hierarchies using language. *arXiv preprint arXiv:2403.01823*, 2024.

Black, K., Brown, N., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Shi, L. X., Tanner, J., Vuong, Q., Walling, A., Wang, H., and Zhilinsky, U. π_0:

A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

Black, K., Brown, N., Darpinian, J., Dhabalia, K., Driess, D., Esmail, A., Equi, M., Finn, C., Fusai, N., Galliker, M. Y., Ghosh, D., Groom, L., Hausman, K., Ichter, B., Jakubczak, S., Jones, T., Ke, L., LeBlanc, D., Levine, S., Li-Bell, A., Mothukuri, M., Nair, S., Pertsch, K., Ren, A. Z., Shi, L. X., Smith, L., Springenberg, J. T., Stachowicz, K., Tanner, J., Vuong, Q., Walke, H., Walling, A., Wang, H., Yu, L., and Zhilinsky, U. $\pi_{0.5}$: a vision-language-action model with open-world generalization. *arXiv preprint arXiv:2504.16054*, 2025.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Dabis, J., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jackson, T., Jesmonth, S., Joshi, N. J., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, K.-H., Levine, S., Lu, Y., Malla, U., Manjunath, D., Mordatch, I., Nachum, O., Parada, C., Peralta, J., Perez, E., Pertsch, K., Quiambao, J., Rao, K., Ryoo, M., Salazar, G., Sanketi, P., Sayed, K., Singh, J., Sontakke, S., Stone, A., Tan, C., Tran, H., Vanhoucke, V., Vega, S., Vuong, Q., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. RT-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., Florence, P., Fu, C., Arenas, M. G., Gopalakrishnan, K., Han, K., Hausman, K., Herzog, A., Hsu, J., Ichter, B., Irpan, A., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Leal, I., Lee, L., Lee, T.-W. E., Levine, S., Lu, Y., Michalewski, H., Mordatch, I., Pertsch, K., Rao, K., Reymann, K., Ryoo, M., Salazar, G., Sanketi, P., Sermanet, P., Singh, J., Singh, A., Soricut, R., Tran, H., Vanhoucke, V., Vuong, Q., Wahid, A., Welker, S., Wohlhart, P., Wu, J., Xia, F., Xiao, T., Xu, P., Xu, S., Yu, T., and Zitkovich, B. RT-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Google DeepMind. Gemini 3 pro model card. https://storage.googleapis.com/deepmind-media/Model-Cards/Gemini-3-Pro-Model-Card.pdf, 2025.

Han, S., Qiu, B., Liao, Y., Huang, S., Gao, C., Yan, S., and Liu, S. RoboCerebra: A large-scale benchmark for long-horizon robotic manipulation evaluation. *arXiv preprint arXiv:2506.06677*, 2025.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations (ICLR)*. OpenReview.net, 2022.

Huang, W., Wang, C., Zhang, R., Li, Y., Wu, J., and Fei-Fei, L. Voxposer: Composable 3D value maps for robotic manipulation with language models. In *Proceedings of the 7th Conference on Robot Learning (CoRL)*, volume 229 of *Proceedings of Machine Learning Research*. PMLR, 2023.

Kim, M. J., Pertsch, K., Karamcheti, S., Xiao, T., Balakrishna, A., Nair, S., Rafailov, R., Foster, E., Lam, G., Sanketi, P., Vuong, Q., Kollar, T., Burchfiel, B., Tedrake, R., Sadigh, D., Levine, S., Liang, P., and Finn, C. OpenVLA: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

Liang, J., Huang, W., Xia, F., Xu, P., Hausman, K., Ichter, B., Florence, P., and Zeng, A. Code as policies: Language model programs for embodied control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 9493–9500. IEEE, 2023.

Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., and Stone, P. LIBERO: Benchmarking knowledge transfer for lifelong robot learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 44776–44791. Curran Associates, Inc., 2023.

Mees, O., Hermann, L., Rosete-Beas, E., and Burgard, W. CALVIN: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters*, 7(3):7327–7334, 2022. doi: 10.1109/LRA.2022.3180108.

Octo Model Team, Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Kreiman, T., Xu, C., Luo, J., Tan, Y., Chen, L. Y., Sanketi, P., Vuong, Q., Xiao, T., Sadigh, D., Finn, C., and Levine, S. Octo: An open-source generalist robot policy. In *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In Gordon, G., Dunson, D., and Dudík, M. (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

Shi, L. X., Ichter, B., Equi, M., Ke, L., Pertsch, K., Vuong, Q., Tanner, J., Walling, A., Wang, H., Fusai, N., Li-Bell, A., Driess, D., Groom, L., Levine, S., and Finn, C. Hi robot: Open-ended instruction following with hierarchical vision-language-action models. In *Proceedings of the*

42nd International Conference on Machine Learning, volume 267 of *Proceedings of Machine Learning Research*, Vancouver, Canada, 2025. PMLR.

Shukor, M., Aubakirova, D., Capuano, F., Kooijmans, P., Palma, S., Zouitine, A., Aractingi, M., Pascal, C., Russi, M., Marafioti, A., Alibert, S., Cord, M., Wolf, T., and Cadene, R. SmolVLA: A vision-language-action model for affordable and efficient robotics. *arXiv preprint arXiv:2506.01844*, 2025.

Singh, I., Blukis, V., Mousavian, A., Goyal, A., Xu, D., Tremblay, J., Fox, D., Thomason, J., and Garg, A. ProgPrompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 11523–11530. IEEE, 2023. doi: 10.1109/ICRA48891.2023.10161317.

Todorov, E., Erez, T., and Tassa, Y. MuJoCo: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

Wen, J., Zhu, Y., Li, J., Zhu, M., Tang, Z., Wu, K., Xu, Z., Liu, N., Cheng, R., Shen, C., Peng, Y., Feng, F., and Tang, J. TinyVLA: Towards fast, data-efficient vision-language-action models for robotic manipulation. *IEEE Robotics and Automation Letters*, 2025. arXiv preprint arXiv:2409.12514.

Zhang, J., Guo, Y., Chen, X., Wang, Y.-J., Hu, Y., Shi, C., and Chen, J. HiRT: Enhancing robotic control with hierarchical robot transformers. *arXiv preprint arXiv:2410.05273*, 2024.

Zhu, Y., Wong, J., Mandlekar, A., Martín-Martín, R., Joshi, A., Lin, K., Maddukuri, A., Nasiriany, S., and Zhu, Y. robosuite: A modular simulation framework and benchmark for robot learning. In *arXiv preprint arXiv:2009.12293*, 2020.