

---

# LITE VLA: EFFICIENT VISION-LANGUAGE-ACTION CONTROL ON CPU-BOUND EDGE ROBOTS

---

A PREPRINT

**Justin Williams<sup>1</sup>, Kishor Datta Gupta<sup>2</sup>, Roy George<sup>3</sup>, and Mrinmoy Sarkar<sup>4</sup>**

<sup>1,2,3</sup>Department of Cyber-Physical Systems, Clark Atlanta University, Atlanta, GA, USA

<sup>4</sup>Siemens Corporation, Princeton, NJ, USA

<sup>1</sup>[justin.williams1@students.cau.edu](mailto:justin.williams1@students.cau.edu), <sup>2</sup>[kgupta@cau.edu](mailto:kgupta@cau.edu), <sup>3</sup>[rgeorge@cau.edu](mailto:rgeorge@cau.edu), <sup>4</sup>[mrinmoy.sarkar@siemens.com](mailto:mrinmoy.sarkar@siemens.com)

November 11, 2025

## ABSTRACT

The deployment of artificial intelligence models at the edge is increasingly critical for autonomous robots operating in GPS-denied environments where local, resource-efficient reasoning is essential. This work demonstrates the feasibility of deploying small Vision-Language Models (VLMs) on mobile robots to achieve real-time scene understanding and reasoning under strict computational constraints. Unlike prior approaches that separate perception from mobility, the proposed framework enables simultaneous movement and reasoning in dynamic environments using only on-board hardware. The system integrates a compact VLM with multimodal perception to perform contextual interpretation directly on embedded hardware, eliminating reliance on cloud connectivity. Experimental validation highlights the balance between computational efficiency, task accuracy, and system responsiveness. Implementation on a mobile robot confirms one of the first successful deployments of small VLMs for concurrent reasoning and mobility at the edge. This work establishes a foundation for scalable, assured autonomy in applications such as service robotics, disaster response, and defense operations.

**Keywords** VLMs, Edge AI, Robotics, Multimodal, Quantization, LoRA, QLoRA

## 1 Introduction

The rise of large multimodal models such as PaLM-E, RT-2, and SayCan has transformed visuomotor reasoning, enabling robots to interpret complex environments and translate high-level goals into actions. However, these systems depend heavily on cloud-based computation and extensive pretraining resources, which makes them impractical for deployment in edge scenarios where connectivity, power, and compute are limited. Field robots operating in disaster zones, underground facilities, or defense missions require fully self-contained intelligence that can reason and act locally without relying on external infrastructure.

To address these limitations, this paper presents **LiteVLA**, a lightweight Vision-Language-Action (VLA) prototype designed for on-device inference and control. LiteVLA extends the capabilities of small multimodal transformers to real-time visuomotor translation by directly mapping visual observations to structured motion commands. Through parameter-efficient fine-tuning using LoRA and 4-bit quantization, the framework achieves efficient on-board operation on resource-constrained hardware such as the Raspberry Pi 4. Unlike conventional pipelines that decouple perception, planning, and control, LiteVLA unifies these processes within a single multimodal reasoning loop integrated with ROS 2 (see Fig. 1).

By leveraging NF4 quantization and the `llama-cpp` runtime, the proposed LiteVLA implementation pioneers the CPU-only deployment path, achieving functional asynchronous visuomotor control on the low-cost Raspberry Pi 4. This represents a novel deployment strategy not demonstrated by prior GPU-centric VLA frameworks such as SmolVLA by Shukor et al. [22], whose work focused primarily on static robotic arms. Beyond proving technical

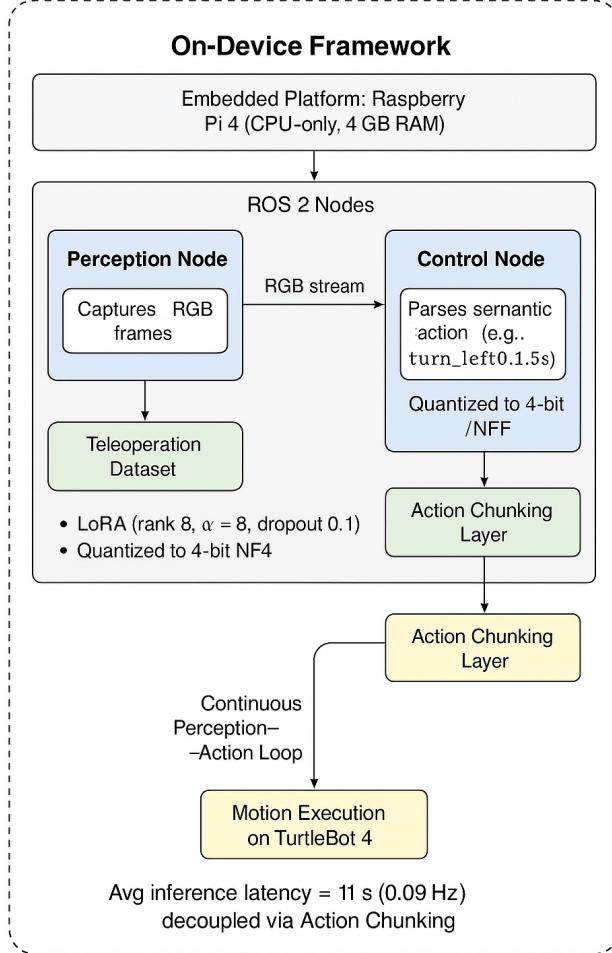


Figure 1: Overview of the LiteVLA system architecture integrating on-device inference, ROS 2 control, and motor actuation.

feasibility, this work establishes a scalable methodology for deploying generalist robot intelligence under strict computational budgets.

To summarize our contributions are:

- **CPU-only on-device VLA.** We integrate a GGUF-quantized Vision–Language–Action policy with ROS 2 for fully CPU-based inference on a Raspberry Pi 4 / TurtleBot 4, achieving asynchronous control via Action Chunking and a baseline of  $\sim 11.1$  s per query (0.09 Hz) (Sec. VII, p. 6).
- **Parameter-efficient adaptation.** We fine-tune a compact SmolVLM backbone using LoRA (rank 8,  $\alpha=8$ , dropout 0.1) to specialize visuomotor translation under tight memory/compute budgets (Alg. 1; Sec. III, pp. 2–3).
- **Edge quantization for stability/efficiency.** With 4-bit NF4 quantization and a hybrid-precision design (NF4 backbone + FP32 projection head), we realize  $\sim 75\%$  lower memory use and up to  $9\times$  faster inference than the FP32 LiteVLA baseline, while preserving stable outputs (Sec. III; Table I, p. 4).
- **End-to-end ROS 2 pipeline.** We unify perception, reasoning, and control in a single loop that maps RGB frames to structured motion commands, implemented with llama-cpp runtime and ROS 2 nodes (Fig. 1, p. 1; Sec. III–IV).
- **Scalable roadmap.** We outline a six-phase EDGE-VLA-ROADMAP from ground robots to drones, multi-agent coordination, multimodal grounding, continual/RL, and federated safety (Fig. 3, p. 4; Sec. VI).

The rest of this paper is organized as follows. Section 2 reviews prior work on VLA and edge AI. Section 3 describes the LiteVLA architecture, fine-tuning, and quantization. Section 4 presents experiments and performance results. Section 5 discusses limitations and future extensions. Finally, Section 6 concludes the paper.

## 2 Related Work

### 2.1 Visual-Language-Action Systems

Large-scale multimodal systems such as PaLM-E, SayCan, and RT-2 have shown that unified language-conditioned reasoning enables robots to follow natural language commands and execute complex manipulation tasks. However, these approaches rely heavily on cloud-based computation and high-end GPUs, making them impractical for resource-limited or field-deployed robots. SMolVLA by Shukor et al. [22] introduced a small and efficient vision-language-action framework designed for community-driven robotic experimentation. It demonstrated that compact multimodal transformers could achieve competitive visuomotor reasoning performance while running on consumer-grade GPUs or CPUs. Nonetheless, its validation was limited to static robotic arms and asynchronous control rather than real-time mobile autonomy. The present work extends this paradigm to mobile robots by integrating a quantized VLA directly within ROS 2 for on-device inference and control.

Beyond LiteVLA, several lines of work pushed the idea of language-conditioned action at scale. SayCan combines a large language model with an affordance/value function to constrain language-proposed steps to what a robot can execute, enabling long-horizon reasoning on mobile manipulators [6]. PaLM-E further treats embodiment as an additional modality, demonstrating that a single large multimodal language model can reason across vision, proprioception, and language [7]. Similarly, RT-1 and RT-2 scale robotics transformers to web-scale data, translating high-level instructions into robot actions [8, 9]. Open-source initiatives such as OpenVLA and Octo [10, 11] advance generalist VLA policies, while works like CLIPort and VIMA [12, 13] show robust language-conditioned visuomotor control with structured spatial reasoning.

### 2.2 On-board Vision-Language Navigation

Recent efforts have focused on improving inference efficiency to enable autonomous navigation directly on edge devices. Chen et al. [14] introduced GRaD-Nav++, a lightweight framework for drone navigation with differentiable reinforcement learning and Gaussian Splatting-based scene representations. Wen et al. [15] proposed TinyVLA, which optimized token efficiency and reduced inference latency for embedded robotic deployment. Complementarily, Budzianowski et al. [16] developed EdgeVLA, which minimizes end-to-end latency through hybrid quantization and communication-aware execution. These approaches emphasize a trade-off between computational load and accuracy; however, LiteVLA differs by achieving fully CPU-based 4-bit inference and real-time control within ROS 2 on a Raspberry Pi 4.

### 2.3 Edge AI and Quantization

Advances in efficient model compression and parameter-efficient fine-tuning (PEFT) have accelerated the deployment of large models on limited hardware. Hu et al. [17] introduced LoRA, a low-rank adaptation method that fine-tunes transformer models by injecting trainable low-rank matrices into frozen layers. Dettmers et al. [18] extended this approach with QLoRA, incorporating NF4 quantization and double quantization for 4-bit training without major accuracy loss. LoRA+ [19] further refined this optimization for faster convergence. Marafioti et al. [20] proposed SmolVLM, a compact multimodal transformer family using GGUF and NF4 formats, while Li et al. [21] introduced token-efficient architectures for edge deployment. These techniques collectively enable on-device inference for vision-language-action systems, directly inspiring the LiteVLA design.

## 3 Implementation Details

The system comprises three primary nodes: dataset capture, multimodal inference, and ROS 2 control. For Data Pipeline purpose, we constructs, preprocesses, and adapts a lightweight Vision–Language–Action (VLA) model for on-device reasoning under limited computational resources (as illustrated in Algorithm 1). A teleoperation phase generates a dataset  $\mathcal{T} = \{(I_t, a_t)\}_{t=1}^N$ , where  $I_t$  is an RGB frame and  $a_t$  the corresponding action vector. Frames are temporally synchronized with actions using timestamps  $\tau_t$ , forming  $\mathcal{D} = \{(I_t, a_t, \tau_t) \mid \tau_t^I \approx \tau_t^a\}$  to ensure one-to-one visuomotor alignment. Each image is standardized via  $\hat{I}_t = \text{augment}(\text{normalize}(\text{resize}(I_t, 224 \times 224)))$ , where  $\text{normalize}(I_t) = (I_t - \mu)/\sigma$  and  $\text{augment}(\cdot)$  applies random flips for robustness. The dataset is parti-

**Algorithm 1:** LiteVLA Data Preparation and Fine-Tuning Pipeline

---

**Input:** Teleoperation trajectories  $\mathcal{T} = \{(I_t, a_t)\}_{t=1}^N$   
**Output:** Trained LiteVLA checkpoint  $\Theta^*$

- 1 **Initialization:**
- 2 Load pretrained multimodal parameters  $\Theta_0$  of SmolVLM backbone;
- 3 Initialize LoRA adapter parameters  $\Delta_{\text{LoRA}}$  with rank  $r = 8$ , scaling  $\alpha = 8$ , and dropout  $p = 0.1$ ;
- 4 **Step 1: Data Acquisition:**
- 5 Capture teleoperation stream of RGB images  $\{I_t\}$  and corresponding action commands  $\{a_t\}$ ;
- 6 Store data as time series  $\mathcal{T} = \{(I_t, a_t, \tau_t)\}$  with timestamps  $\tau_t$ ;
- 7 **Step 2: Synchronization:**
- 8 Align each image  $I_t$  with its action  $a_t$  s.t.  $\tau_t^I \approx \tau_t^a$ ;
- 9

$$\mathcal{D} = \{(I_t, a_t, \tau_t) \mid \tau_t^I \approx \tau_t^a\}$$

- 10 **Step 3: Preprocessing:**
- 11 For each  $I_t \in \mathcal{D}$ :
- 12      $\hat{I}_t \leftarrow \text{augment}(\text{normalize}(\text{resize}(I_t, 224 \times 224)))$
- 13     where  $\text{normalize}(I_t) = \frac{I_t - \mu}{\sigma}$ ;
- 14 Store preprocessed dataset  $\hat{\mathcal{D}} = \{(\hat{I}_t, a_t)\}$ ;
- 15 **Step 4: Dataset Partition:**
- 16 Split  $\hat{\mathcal{D}}$  into training and validation subsets with ratio 0.85 : 0.15;
- 17

$$\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}} \leftarrow \text{split}(\hat{\mathcal{D}}, 0.85)$$

- 18 **Step 5: LoRA Fine-Tuning:**
  - 19 Update model parameters via minimization:
- $$\Theta^* = \arg \min_{\Theta_0, \Delta_{\text{LoRA}}} \mathbb{E}_{(I_t, a_t) \sim \mathcal{D}_{\text{train}}} [\mathcal{L}(f_{\Theta_0 + \Delta_{\text{LoRA}}}(I_t), a_t)]$$
- subject to  $\text{rank}(\Delta_{\text{LoRA}}) = 8$ ,  $\alpha = 8$ , dropout = 0.1;
- 20 **Step 6: Checkpoint Generation:**
  - 21 Store optimized parameters as final checkpoint:

$$\text{Save: } \Theta^* = \{\Theta_0, \Delta_{\text{LoRA}}\}$$

- 22 **Return:** Trained LiteVLA checkpoint  $\Theta^*$ ;
- 

tioned into training and validation sets ( $\mathcal{D}_{\text{train}} : \mathcal{D}_{\text{val}} = 0.85 : 0.15$ ) to ensure statistical independence. A Low-Rank Adaptation (LoRA) scheme fine-tunes the pretrained multimodal backbone (SmolVLM) by minimizing  $\mathbb{E}_{(I_t, a_t) \sim \mathcal{D}_{\text{train}}} [\mathcal{L}(f_{\Theta_0 + \Delta_{\text{LoRA}}}(I_t), a_t)]$  subject to  $\text{rank}(\Delta_{\text{LoRA}}) = 8$ ,  $\alpha = 8$ , and  $p_{\text{dropout}} = 0.1$ , yielding optimized parameters  $\Theta^* = \{\Theta_0, \Delta_{\text{LoRA}}\}$  as the deployable LiteVLA checkpoint for efficient on-device inference on low-power hardware such as the Raspberry Pi 4.

Figure 2 illustrates the comparison between an FP32 full-precision model and a 4-bit quantized model (NF4). The FP32 model exhibits high memory usage leading to slower inference and higher power consumption, while quantization converts it into a lightweight NF4 model achieving approximately 75% lower memory usage, faster inference, and reduced power requirements.

This lightweight adaptation strategy enables the vision-language backbone to specialize in translating RGB inputs into robot motion commands while maintaining minimal computational overhead. The fine-tuned model is then quantized to 4-bit NF4 format for deployment, enabling CPU-only inference through the `llama-cpp-python` runtime with reduced memory usage and latency.

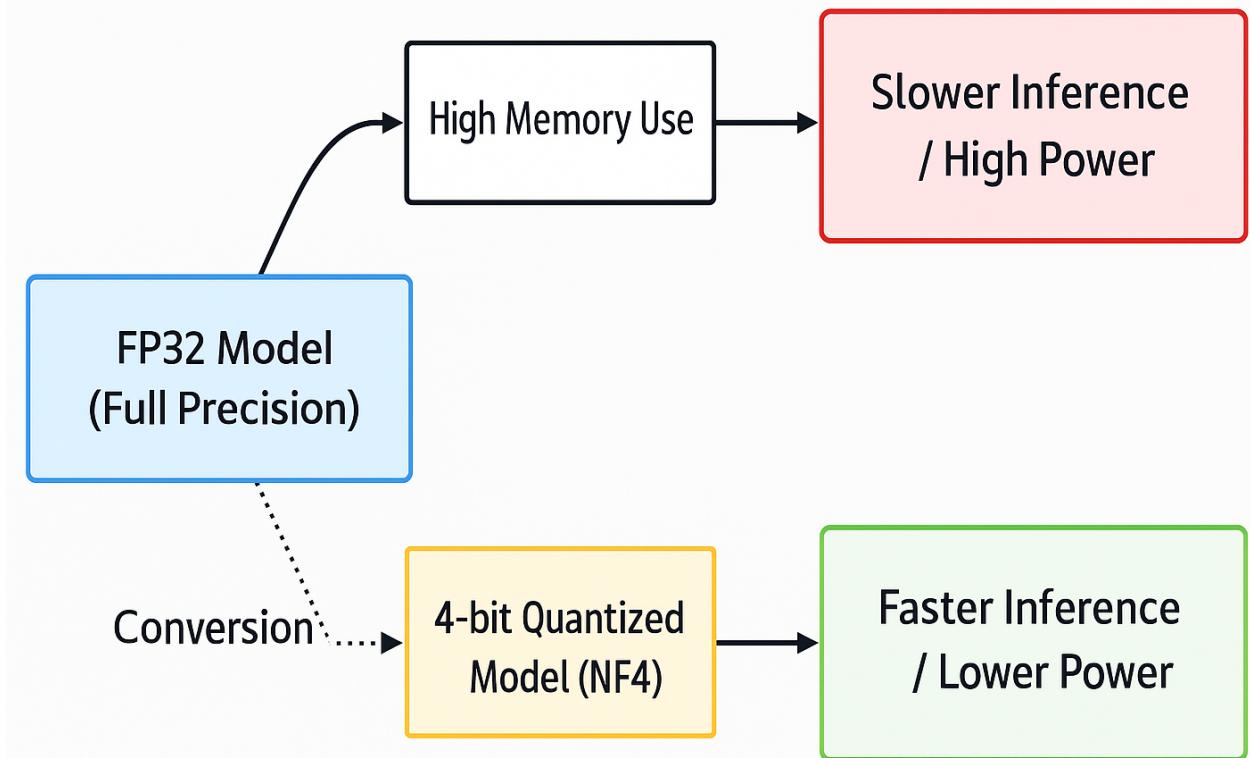


Figure 2: LiteVLA architecture highlighting LoRA-adapted layers and NF4 quantization workflow for CPU deployment.

## 4 Experiment and Performance Analysis

### 4.1 Dataset and Training

The dataset used for training consists of 15,083 image–action pairs collected through manual teleoperation of a Turtle-Bot 4. Each RGB frame is synchronized with a JSON label that encodes the robot’s linear and angular velocities along with the corresponding timestamp. The data capture node recorded continuous control sessions to provide a diverse set of visuomotor examples across indoor environments. Although more than 60,000 frames were collected in total, only a subset of approximately 13,000 images was used for initial experimentation to verify on-device inference. This subset includes 1,152 samples of backward motion and balanced distributions of forward, left, right, and stop commands, with roughly 2,990 samples each. Every image in the dataset is paired with a corresponding action label describing the intended motion command, forming a direct mapping from perception to control.

The dataset was divided into training and validation partitions using an 85:15 ratio. Images were resized to 224×224 pixels, normalized, and augmented through random horizontal flips to improve generalization. The resulting dataset was used to fine-tune the multimodal transformer using a LoRA configuration with rank 8,  $\alpha = 8$ , and dropout 0.1 for approximately two epochs on a CPU backend.

### 4.2 Model Architecture and Quantization

The fine-tuning process employs a Low-Rank Adaptation (LoRA) configuration with rank  $r = 8$ , scaling factor  $\alpha = 8$ , and dropout of 0.1. The LoRA adapters are applied to the transformer’s projection layers, including query, key, value, output, and gating modules. This configuration limits the number of additional parameters while allowing efficient task-specific adaptation. The scaling factor controls the influence of LoRA weights relative to the base model, while dropout introduces regularization to prevent overfitting. By restricting updates to key transformer layers, the model efficiently learns visuomotor associations without modifying the full parameter set.

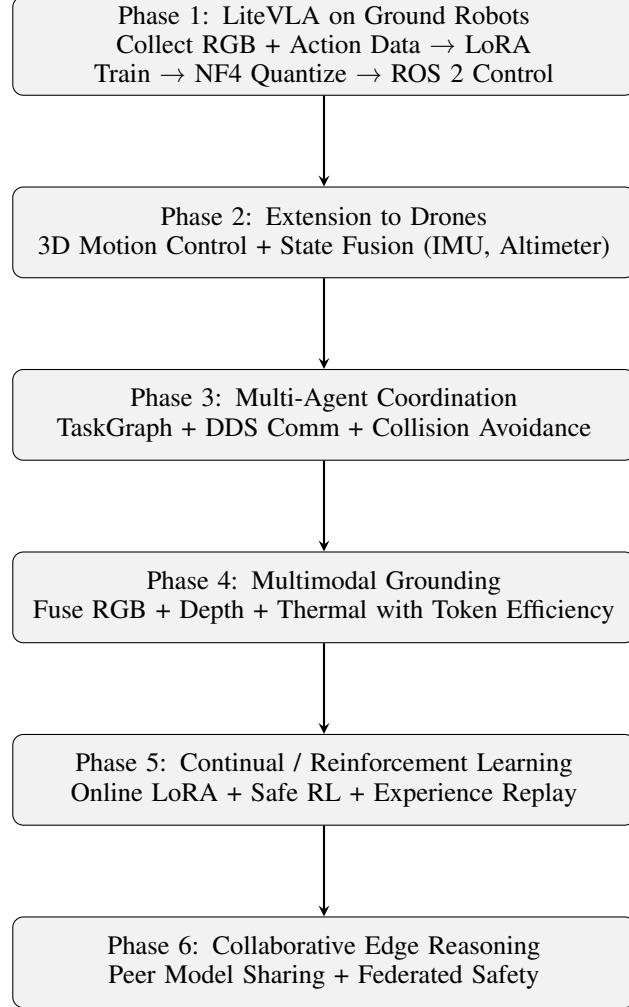


Figure 3: Simplified EDGE-VLA-ROADMAP pipeline showing six evolutionary phases from single-agent LiteVLA deployment to collaborative edge reasoning.

### 4.3 ROS 2 Integration and Runtime Performance

LiteVLA is deployed within the ROS 2 middleware to enable direct visuomotor inference on the TurtleBot 4 platform. During operation, the inference node continuously captures RGB frames from the onboard camera and processes them through the quantized vision-language-action model. Each processed frame produces a semantic action string such as `forward_0.2_3.0s` or `turn_left_0.1_2.5s`, which is parsed and republished as a `geometry_msgs/Twist` command to the robot's motion topic (see Fig. 3).

The system achieves an average per-query VLA inference latency ( $L$ ) of approximately 11 seconds on the quad-core Raspberry Pi 4 CPU, corresponding to 0.09 Hz. This latency defines the maximum sustainable CPU-bound throughput for the 4-bit VLA. The VLA outputs are processed using an Action Chunking mechanism within ROS 2, ensuring that the TurtleBot's low-level motion controller maintains predictable control while high-level reasoning operates asynchronously.

SmolVLM [20] serves as the foundational architecture for LiteVLA. It is a compact multimodal transformer family ranging from 256 million to 2.2 billion parameters, designed for energy-efficient vision-language reasoning. Pixel-shuffle tokenization divides each image into sub-images that are spatially rearranged to reduce the number of visual tokens while preserving essential spatial information. The smallest configuration, SmolVLM-256M, operates with less than 1 GB of memory during inference, making it suitable for embedded platforms such as the Raspberry Pi 4.

## 4.4 Comparison

Table 1 compares inference latency and configuration across several LiteVLA variants on a Raspberry Pi 4. The *SmolVLM-256* baseline (FP32) achieves approximately 11 s per inference, serving as the reference for latency and stability. In contrast, the unquantized *LiteVLA* (FP32) model exhibits a substantial slowdown, requiring nearly 18 minutes per forward pass due to its larger parameter set and higher computational demand. Applying hybrid quantization—using a 4-bit NF4 backbone while preserving the projection head in FP32—yields a drastic performance improvement, reducing latency to around 2 minutes per inference, a 9 $\times$  speedup over the FP32 LiteVLA baseline. This configuration also maintains stable action predictions and consistent motion commands, representing the best trade-off between efficiency and reliability. A fully quantized NF4 version (*LiteVLA 4b*) achieves the fastest execution time ( $\sim$ 1.5 min), but exhibits degraded and unstable outputs due to precision loss in the projection head, occasionally leading to hallucinated or inconsistent control signals. Overall, these results validate that hybrid precision offers the optimal configuration for CPU-only edge deployment, balancing speed, memory footprint, and output stability on low-power embedded hardware.

## 5 Threats to Validity

While LiteVLA demonstrates the feasibility of fully on-device vision-language-action reasoning, several factors may affect the validity and generalizability of our results.

### 5.1 Computational Latency and Quantization Effects

Inference latency remains bounded by the limited compute of the Raspberry Pi 4 CPU. The unquantized FP32 model requires approximately 18 minutes per forward pass, making real-time control infeasible. Applying 4-bit NF4 quantization (while preserving the projection head in FP32) reduces latency to approximately 2 minutes per inference, achieving nearly a 9 $\times$  improvement. However, quantizing the projection head induces severe degradation, producing unstable or hallucinatory outputs. Consequently, the projection head remains in FP32 precision in all experiments.

### 5.2 Model Compression and Representational Fidelity

Although quantization reduces memory consumption by roughly 75%, it also compresses representational granularity, which occasionally leads to coarse-grained action predictions. The hybrid-precision configuration (NF4 backbone + FP32 head) mitigates this degradation but remains sensitive to thermal throttling and memory bandwidth limitations inherent to embedded hardware.

### 5.3 Experimental Hardware Constraints

All on-device experiments were conducted on a Raspberry Pi 4 equipped with 4 GB of RAM and a quad-core ARM Cortex-A72 CPU running at 1.5 GHz. The limited CPU throughput and absence of hardware acceleration (such as GPU or NPU support) significantly constrain both inference and preprocessing throughput. In particular, simultaneous RGB frame capture, normalization, and quantized model execution compete for the same CPU cores, leading to noticeable latency spikes under sustained operation.

Thermal throttling further exacerbates these constraints, as prolonged inference at near-maximum utilization causes CPU frequency scaling and inconsistent timing between reasoning and actuation. Additionally, memory bandwidth limitations restrict batch sizes and context length, preventing parallel frame evaluation and constraining temporal reasoning within a single frame horizon. These combined factors indicate that the primary performance bottlenecks in LiteVLA stem from the compute and thermal limits of the Raspberry Pi 4 hardware rather than from inefficiencies in the model architecture itself.

### 5.4 Dataset and Task Variability

The training dataset, collected via manual teleoperation, may not capture the full diversity of real-world navigation conditions. Consequently, model generalization to unstructured or outdoor environments remains an open question for future work.

**Table 1: Inference Latency and Configuration Comparison**

Model	Prec.	Device	Latency	Notes
SmolVLM-256	FP32	RPi 4	~11 s	Baseline
LiteVLA (Ours)	FP32	RPi 4	~18 min	Unquantized
LiteVLA (Hybrid)	4b + FP32	RPi 4	~2 min	Stable; 9× faster
LiteVLA (NF4)	4b	RPi 4	~1.5 min	Unstable output

## 6 Discussion and Future Work

The success of LiteVLA in performing perception-to-action reasoning on a compact robotic platform demonstrates the feasibility of deploying language-grounded intelligence at the edge. Future work will extend this capability beyond ground robots to aerial systems, using a drone as the next platform for experimentation. A drone equipped with cameras and onboard processing will enable three-dimensional perception and movement, expanding the applicability of the framework to exploration, surveillance, and mapping tasks.

An important direction is enabling autonomous navigation in previously unseen environments after the model has been trained in familiar ones. This transition from structured to unstructured domains will require the integration of continual learning and self-adaptation, allowing the model to refine its visual and spatial understanding during deployment. Through this, LiteVLA can evolve from reactive control to proactive reasoning, maintaining robustness under dynamic and uncertain conditions.

The EDGE-VLA-ROADMAP diagram (Fig. 3) visually summarizes the evolutionary progression of the LiteVLA framework from single-agent ground-robot operation to collaborative multi-agent intelligence at the edge. The figure is organized into six phases, each subdivided into three sub-phases that collectively describe how lightweight vision-language-action (VLA) reasoning evolves into distributed, self-adaptive autonomy. It begins with Phase 1, where the LiteVLA model is trained on teleoperated RGB-action datasets, fine-tuned using Low-Rank Adaptation (LoRA,  $r = 8$ ,  $\alpha = 8$ , dropout = 0.1), and deployed on a Raspberry Pi 4 with NF4 quantization integrated through ROS 2 nodes. Phase 2 extends these capabilities to aerial drones by introducing three-dimensional actuation, IMU-altimeter sensor fusion, and domain randomization for robust flight control. Phase 3 enables multi-agent coordination via a TaskGraph structure and ROS 2 DDS communication for shared state awareness, ensuring collision avoidance and graceful degradation when connectivity weakens.

The later stages focus on increasing adaptability and collaboration. Phase 4 introduces multimodal grounding by fusing RGB, depth, and thermal data through pixel-shuffle tokenization and quantization-aware fusion to improve perception under challenging conditions. Phase 5 implements continual and reinforcement learning using on-device experience replay, online LoRA updates, and safe PPO-based optimization to refine policies while maintaining operational safety. Finally, Phase 6 establishes collaborative edge reasoning, where robots exchange LoRA deltas through peer-to-peer knowledge gossip, perform weighted model merging with rollback validation, and enforce federated safety protocols without sharing raw sensor data. Together, these interconnected stages define a full-spectrum roadmap for scalable, self-improving, and privacy-preserving edge autonomy in multimodal robotic systems.

## 7 Conclusions

This work establishes a complete end-to-end framework for vision-language-action learning and deployment on resource-constrained robotic platforms. We present the first integration of a GGUF-quantized VLA policy within a ROS environment, enabling fully CPU-based inference without GPUs. This CPU-centric deployment methodology demonstrates that multimodal reasoning and control can operate locally, directly on low-power educational robots such as the TurtleBot 4 equipped with a Raspberry Pi 4 containing only 4 GB of RAM.

The system demonstrated a per-query VLA inference latency of approximately 11.11 seconds, equivalent to 0.09 Hz. This defines a measurable performance baseline for edge-based reasoning. Importantly, this does not reflect the robot’s control rate—the ROS 2 Action Chunking mechanism allows continuous low-level control while reasoning operates asynchronously.

## Acknowledgment

This research was supported by the U.S. Department of Education, NASA USRC and NASA MPLAN awards.

## References

- [1] Q. Chen, N. Gao, S. Huang, J. E. Low, T. Chen, J. Sun, and M. Schwager, “GRaD-Nav++: Vision-Language Model Enabled Visual Drone Navigation with Gaussian Radiance Fields and Differentiable Dynamics,” *arXiv preprint arXiv:2506.14009*, 2025.
- [2] A. Marafioti, O. Zohar, M. Farré, *et al.*, “SmolVLM: Redefining Small and Efficient Multimodal Models,” *arXiv preprint arXiv:2504.05299*, 2025.
- [3] M. Shukor, D. Aubakirova, F. Capuano, *et al.*, “SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics,” *arXiv preprint arXiv:2506.01844*, 2025.
- [4] P. Budzianowski, W. Maa, M. Freed, *et al.*, “EdgeVLA: Efficient Vision-Language-Action Models,” *arXiv preprint arXiv:2507.14049v1*, 2025.
- [5] J. Li, R. Patel, A. Fedorov, *et al.*, “TinyVLA: Lightweight Vision-Language-Action Models for Edge Robotics,” *arXiv preprint arXiv:2504.19277*, 2025.
- [6] M. Ahn, A. Brohan, N. Brown, *et al.*, “Do As I Can, Not As I Say: Grounding Language in Robotic Affordances,” in *Proceedings of the 6th Conference on Robot Learning (CoRL)*, 2022.
- [7] D. Driess, F. Xia, M. S. Sajjadi, *et al.*, “PaLM-E: An Embodied Multimodal Language Model,” *arXiv preprint arXiv:2303.03378*, 2023.
- [8] A. Brohan, N. Brown, J. Ichter, *et al.*, “RT-1: Robotics Transformer for Real-World Control at Scale,” *arXiv preprint arXiv:2212.06817*, 2022.
- [9] A. Brohan, C. Chebotar, and N. Brown, “RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control,” *arXiv preprint arXiv:2307.15818*, 2023.
- [10] J. Kim, D. Choi, *et al.*, “OpenVLA: Open-Source Generalist Vision-Language-Action Model for Robotics,” *arXiv preprint arXiv:2403.08265*, 2024.
- [11] Z. Yang, S. Lin, *et al.*, “Octo: Open-Vocabulary Object Manipulation with Pretrained Vision-Language Models,” *arXiv preprint arXiv:2404.11012*, 2024.
- [12] M. Shridhar, C. Manuelli, and D. Fox, “CLIPort: What and Where Pathways for Robotic Manipulation,” in *Proceedings of the Conference on Robot Learning (CoRL)*, 2022.
- [13] Y. Jiang, A. Xu, C. Gupta, *et al.*, “VIMA: General Robot Manipulation with Multimodal Prompts,” *arXiv preprint arXiv:2210.03094*, 2022.
- [14] Q. Chen, N. Gao, and S. Huang, “GRaD-Nav++: Vision-Language Model Enabled Visual Drone Navigation with Gaussian Radiance Fields and Differentiable Dynamics,” *arXiv preprint arXiv:2506.14009*, 2025.
- [15] S. Wen, R. Zhao, and A. Fedorov, “TinyVLA: Lightweight Vision-Language-Action Models for Edge Robotics,” *arXiv preprint arXiv:2404.19277*, 2024.
- [16] P. Budzianowski, W. Maa, and M. Freed, “EdgeVLA: Efficient Vision-Language-Action Models for Edge Devices,” *arXiv preprint arXiv:2507.14049*, 2025.
- [17] E. J. Hu, Y. Shen, P. Wallis, *et al.*, “LoRA: Low-Rank Adaptation of Large Language Models,” *arXiv preprint arXiv:2106.09685*, 2021.
- [18] T. Dettmers, A. Lewis, S. Shen, *et al.*, “QLoRA: Efficient Finetuning of Quantized LLMs,” *arXiv preprint arXiv:2305.14314*, 2023.
- [19] M. Zhao and H. Wang, “LoRA+: Enhanced Low-Rank Adaptation for Faster Convergence in Fine-Tuning,” *arXiv preprint arXiv:2401.03103*, 2024.
- [20] A. Marafioti, O. Zohar, and M. Farre, “SmolVLM: Redefining Small and Efficient Multimodal Models,” *arXiv preprint arXiv:2504.05299*, 2025.
- [21] J. Li, R. Patel, and A. Fedorov, “TinyVLA: Lightweight Vision-Language-Action Models for Edge Robotics,” *arXiv preprint arXiv:2504.19277*, 2025.
- [22] M. Shukor, D. Aubakirova, and F. Capuano, “SmolVLA: A Vision-Language-Action Model for Affordable and Efficient Robotics,” *arXiv preprint arXiv:2506.01844*, 2025.