PhD Application Take-Home Assignment

Embodied AI Generalization Problem

Santhosh

January 15, 2026

Contents

1 Introduction

1.1 The Generalization Problem in Embodied AI

One of the fundamental challenges in embodied artificial intelligence is **generalization**. Traditional robot learning approaches require task-specific training for every new scenario, making them impractical for real-world deployment where variability is the norm rather than the exception.

Consider a robot trained to pick up a red cup from a specific location. Classical approaches would fail if presented with a blue mug in a different position, even though the underlying manipulation skills are identical. This gap in skill stems from learning narrow, overfitted policies that capture superficial correlations rather than the fundamental principles of manipulation.

The generalization problem manifests in multiple dimensions:

- **Object variation**: Different shapes, sizes, colors, and materials
- **Spatial variation**: Novel configurations and placement locations
- **Task variation**: New instruction phrasings and goal specifications
- **Environmental variation**: Different backgrounds, lighting, and contexts

For this assignment, I focused on **cross-task generalization within robotic manipulation**. Specifically, how vision-language-action (VLA) models can learn manipulation skills that transfer across different spatial configurations and object categories.

1.2 Why This Problem Matters

The inability to generalize severely limits the practical deployment of robotic systems. Industrial robots are typically confined to highly structured environments with fixed object positions and known specifications. Service robots struggle in unorganized environments such as homes and offices, where every room presents unique challenges. The computational cost and human effort required to collect task-specific demonstrations for every scenario makes scaling prohibitively expensive.

Recent breakthroughs in vision-language models (VLMs) have demonstrated that models pretrained on internet-scale data can develop rich, generalizable representations of the visual world and natural language. Vision-language-action models extend this paradigm to robotics, hypothesizing that these pretrained representations can serve as a foundation for manipulation skills that generalize broadly.

1.3 Experimental Constraints

This investigation was conducted under a few resource constraints:

- **Compute platform**: Google Colab with A100 GPU access
- **Budget**: Approximately $20 in compute credits (200 Colab credit points)
- **Time**: All experiments executed within roughly one week
- **Evaluation scope**: Reduced to 5 episodes per task (vs. 50 in original paper) to maintain reasonable runtime

Despite these limitations, the experiments successfully demonstrate reproducibility of state-of-the-art results and validate the effectiveness of inference-time improvements.

2 Identifying State-of-the-Art Method

2.1 Background Research

Before selecting a specific method to replicate, I conducted extensive background research to understand the current state of embodied AI and vision-language-action models. This involved reviewing multiple surveys, tutorials, and foundational papers. Some of the resources I found particularly helpful included:

Embodied AI Fundamentals

- https://encord.com/blog/embodied-ai/
- https://www.nvidia.com/en-us/glossary/embodied-ai/
- https://builtin.com/artificial-intelligence/embodied-ai
- https://lamarr-institute.org/blog/embodied-ai-explained/

Robot Learning Basics

- https://huggingface.co/learn/robotics-course/unit1/1
- https://16-831-s24.github.io/
- https://www.nvidia.com/en-us/learn/learning-path/robotics/
- https://www.geeksforgeeks.org/machine-learning/what-is-policy-in-reinforcement-lea

Vision-Language Models

- https://www.ultralytics.com/blog/understanding-vision-language-models-and-their-ap
- https://www.ibm.com/think/topics/vision-language-models
- https://learnopencv.com/vision-language-action-models-lerobot-policy/
- https://jalammar.github.io/illustrated-transformer/
- https://poloclub.github.io/transformer-explainer/

Imitation Learning

- https://www.nvidia.com/en-us/glossary/imitation-learning/
- https://underactuated.mit.edu/imitation.html
- https://huggingface.co/docs/lerobot/en/il_robots

VLA and Robotics Surveys

- [14] - A Survey on Vision-Language-Action Models for Embodied AI
- [15] - Foundation Models in Robotics: Applications, Challenges, and the Future
- [16] - Toward General-Purpose Robots via Foundation Models
- [17] - A Survey of Imitation Learning

Google DeepMind Blog Posts

- https://deepmind.google/discover/blog/shaping-the-future-of-advanced-robotics/
- https://blog.google/innovation-and-ai/products/google-deepmind-rt2-robotics-vla-mo
- https://deepmind.google/discover/blog/scaling-up-learning-across-many-different-ro

2.2 Why OpenVLA?

After reviewing multiple state-of-the-art approaches in embodied AI, I selected **OpenVLA** [1] as the foundation for this work. The decision was based on several factors:

- **Performance**: Achieves 16.5% improvement over Google's RT-2-X[11] while using 7B parameters vs. 55B
- **Accessibility**: Fully open-source with pretrained weights and fine-tuning code
- **Documentation**: Well-documented codebase with reproducible evaluation scripts
- **Resource efficiency**: Specifically built to be runnable on consumer hardware (vs. requiring data center infrastructure), perfect for this assignment's constraints

Alternative methods considered included **GraspMolmo**[8] and **Octo**[7], but OpenVLA provided the best balance of performance, accessibility, and documentation quality for the available timeframe.

2.3 Architecture Overview

OpenVLA is a 7-billion parameter vision-language-action model that extends pretrained vision-language backbones to output robot actions instead of text tokens. The architecture consists of three key components:

2.3.1 Dual Vision Encoders

Unlike single-encoder approaches, OpenVLA employs two complementary vision systems:

- **SigLIP**[9]: Trained on 400M image-text pairs using contrastive learning, excels at semantic understanding ("what" things are)
- **DINOv2**[10]: Self-supervised encoder trained on images alone, specializes in spatial precision ("where" things are)

The fusion of semantic and spatial representations proved critical for manipulation tasks, where robots need both object recognition and precise 3D localization.

2.3.2 Language-Conditioned Transformer

A transformer backbone (Llama-2-based) processes the concatenated visual features along with natural language task instructions. This enables the model to understand complex commands like "pick up the black bowl next to the ramekin and place it on the plate."

2.3.3 Action Tokenization

Robot actions are continuous 7-DoF vectors (X, Y, Z position + roll, pitch, yaw orientation + gripper state). OpenVLA discretizes each dimension into 256 bins, converting actions into tokens that the language model can predict autoregressively. This allows the model to leverage the transformer's powerful sequence modeling capabilities for action generation.

2.4 Training Methodology

OpenVLA was trained on the **Open X-Embodiment**[12] dataset, which consists of 970,000 real robot demonstrations from 21 institutions covering 22 robot types and 527 distinct skills. The training used behavioral cloning (supervised imitation learning) with a key insight: training for 27 epochs until action prediction accuracy exceeded 95% was critical for downstream performance.

For task-specific adaptation, OpenVLA supports efficient fine-tuning via LoRA (Low-Rank Adaptation), enabling customization on consumer GPUs in hours rather than weeks.

2.5 LIBERO Benchmark

The LIBERO[4] benchmark evaluates knowledge transfer for lifelong robot learning across multiple dimensions:

- **libero_spatial**: Tests generalization to novel spatial configurations
- **libero_object**: Tests generalization to different object categories
- **libero_goal**: Tests understanding of varied instruction phrasings
- **libero_100**: Large-scale evaluation with 100 diverse tasks

Each task suite contains 10 tasks with different initial conditions, testing whether learned manipulation skills transfer across variations. For this study, I focused on `libero_spatial` and `libero_object` as representative benchmarks.

3 Replication Process

3.1 Phase 1: Baseline Validation

The first step was confirming that OpenVLA could be successfully executed in the Google Colab environment. I cloned the official OpenVLA repository and ran their evaluation script on `libero_spatial` with the pretrained checkpoint.
Challenges encountered:

- Environment setup required careful dependency management between PyTorch, Tensorflow, LIBERO and few other dependencies
- CUDA memory constraints on Colab necessitated batch size adjustments
- Display backend issues with rendering required headless execution

Resolution: Successfully generated rollout videos confirming the model executes manipulation behaviors.

3.2 Phase 2: Full Replication from Scratch

To demonstrate deep understanding of the system, I reimplemented the entire evaluation pipeline in a simplified, standalone notebook. This involved:

1. Loading the pretrained OpenVLA model and task-specific fine-tuned weights
2. Implementing the LIBERO environment interface
3. Creating the observation preprocessing pipeline (image resizing, normalization)
4. Implementing the action prediction loop with unnormalization
5. Adding rollout video generation and success tracking

The reimplementation helped me understand some of the architectural parameters mentioned in the paper. These included:

- Image preprocessing must exactly match training (center crop + resize to 224×224)
- Action unnormalization requires task-specific statistics
- Episode termination logic combines task success signals with timeout conditions

3.3 Evaluation Results

I evaluated OpenVLA on 5 tasks from each of two task suites, running 5 episodes per task (25 episodes total per suite). This reduced scale was necessary given Colab time limits but sufficient to validate reproducibility.

3.3.1 libero_spatial Results

Task	Success Rate	Episodes
Pick up the black bowl between the plate and the ramekin and place it on the plate	100%	5/5
Pick up the black bowl next to the ramekin and place it on the plate	60%	3/5
Pick up the black bowl from table center and place it on the plate	80%	4/5
Pick up the black bowl on the cookie box and place it on the plate	60%	3/5
Pick up the black bowl in the top drawer of the wooden cabinet and place it on the plate	40%	2/5
Overall	**68.0%**	**17/25**

Table 1: OpenVLA baseline results on libero_spatial

3.3.2 libero_object Results

Task	Success Rate	Episodes
Pick up the alphabet soup and place it in the basket	80%	4/5
Pick up the cream cheese and place it in the basket	80%	4/5
Pick up the salad dressing and place it in the basket	40%	2/5
Pick up the bbq sauce and place it in the basket	60%	3/5
Pick up the ketchup and place it in the basket	80%	4/5
Overall	**68.0%**	**17/25**

Table 2: OpenVLA baseline results on libero_object

3.4 Comparison to Original Results

The OpenVLA paper reports success rates on LIBERO-Spatial and LIBERO-Object with 50 episodes per task. While direct comparison is limited by our reduced episode count, the 68% success rate aligns reasonably with expectations given the task difficulty and known failure modes (drawer manipulation, precise placement).
Key observation: Both task suites achieved identical 68% success rates, suggesting OpenVLA's generalization capabilities are consistent across spatial and object variations at this performance level.

4 Proposed Ideas

After successfully replicating OpenVLA, I explored inference-time techniques to improve performance without requiring model retraining. The motivation was practical: retraining large VLA models is expensive and time-consuming, while inference-time improvements can be deployed immediately.

4.1 Candidate Approaches

I identified few promising inference-time techniques from recent literature:

4.1.1 Temporal Ensembling

Smooth action trajectories by averaging overlapping predictions across timesteps, reducing noise and improving execution stability (inspired by ACT [3]).

4.1.2 Real-Time Chunking

Treat action chunk boundaries as an "inpainting" problem to eliminate discontinuities between prediction windows (from Physical Intelligence's RTC work[18]).

4.1.3 Warm-Start Denoising

Initialize inference from previous predictions rather than random noise, reducing mode switching and iteration count [19].

4.1.4 VLM-Guided Execution (LITEN)

Use a separate vision-language model to reason about task execution, assess failures, and guide retries, enabling learning at inference time.

4.2 Selection: LITEN

I chose to implement **LITEN (Learning from Inference-Time Execution)** [2] because it addresses a fundamental limitation of current VLA models: they lack the ability to reflect on failures and adjust behavior accordingly.
LITEN's key insight is hierarchical decomposition:

- **High-level VLM**: Reasons about affordances, generates plans, assesses outcomes
- **Low-level VLA**: Executes actions (OpenVLA in our case)
- **Iterative refinement**: Failed attempts inform subsequent retries

This approach is conceptually appealing because it mirrors human problem-solving. If we don't succeed at complex manipulation tasks on the first try, we iterate based on what we observe.

5 Implementation

5.1 LITEN Architecture

I implemented a simplified version of LITEN that captures the core inference-time iteration paradigm while omitting the full VLM-based reasoning pipeline due to API cost and time constraints.

5.1.1 Components Implemented

Retry Mechanism The fundamental contribution of LITEN is allowing multiple attempts at each task:

```
MAX_RETRIES = 3

while not success and attempt < MAX_RETRIES:
    attempt += 1
    success, steps, failure_mode = run_episode(
        env, model, task, initial_state, attempt
    )
```

This simple modification enables the system to recover from transient failures (e.g., slight grasp misalignments, near-miss placements).

Failure Mode Assessment Each execution is assessed based on progress through the task:

```
def assess_execution(success, num_steps, max_steps):
    if success:
        return "SUCCESS"

    progress = num_steps / max_steps

    if progress < 0.2:
        return "EARLY_FAILURE: grasp/targeting"
    elif progress < 0.5:
        return "MID_FAILURE: manipulation"
    elif progress < 0.8:
        return "LATE_FAILURE: placement"
    else:
        return "TIMEOUT: nearly completed"
```

This provides diagnostic information about failure modes and could guide future VLM-based reasoning.

5.1.2 Components Omitted

External VLM Reasoning The full LITEN uses GPT-4V or Claude to analyze task instructions and execution videos, generating affordance hints and refined plans. To keep it simple, I implemented rule-based heuristics instead:

```
def generate_affordance_hints(task_description):
    hints = []

    if "between" in task_description:
        hints.append("Object: Target middle item")

    if "next to" in task_description:
        hints.append("Spatial: Consider proximity")

    # Task-specific heuristics...
    return hints
```

While less sophisticated than VLM-based reasoning, this captures the spirit of affordance-aware execution.

Video-Based Assessment Full LITEN analyzes video frames from failed executions to identify specific failure causes. Our implementation uses step count and success signals as proxies for execution progress.

5.2 Expected vs. Actual Improvements

5.2.1 Hypothesis

Based on LITEN's design principles, I expected improvements through:

- Recovery from transient grasp failures
- Refined placement after near-misses

5.2.2 Results

Task Suite	Baseline	LITEN	Improvement
libero_spatial	68.0%	96.0%	+28.0%
libero_object	68.0%	76.0%	+8.0%
Average	**68.0%**	**86.0%**	**+18.0%**

Table 3: Performance comparison: OpenVLA baseline vs. LITEN with retries

5.2.3 Detailed Analysis: libero_spatial

Task	Baseline	LITEN	Change
Pick up the black bowl between the plate and the ramekin and place it on the plate	100%	100%	–
Pick up the black bowl next to the ramekin and place it on the plate	60%	100%	+40%
Pick up the black bowl from table center and place it on the plate	80%	100%	+20%
Pick up the black bowl on the cookie box and place it on the plate	60%	100%	+40%
Pick up the black bowl in the top drawer of the wooden cabinet and place it on the plate	40%	80%	+40%
Overall	**68%**	**96%**	**+28%**

Table 4: Task-level breakdown for libero_spatial

The drawer manipulation task (80% final) had 1 episode that failed after 3 retry attempts, suggesting this task requires more fundamental capability improvements beyond retry mechanisms.

5.2.4 Detailed Analysis: libero_object

Task	Baseline	LITEN	Change
Pick up the alphabet soup and place it in the basket	80%	80%	–
Pick up the cream cheese and place it in the basket	80%	40%	-40%
Pick up the salad dressing and place it in the basket	40%	80%	+40%
Pick up the bbq sauce and place it in the basket	60%	80%	+20%
Pick up the ketchup and place it in the basket	80%	100%	+20%
Overall	**68%**	**76%**	**+8%**

Table 5: Task-level breakdown for libero_object

Notably, cream cheese performance decreased with retries (-40%). This could suggest that for certain object categories, the additional-retries strategy may introduce instability. Both increase and decrease in performance across tasks could very well be due to the small sample size (5 episodes per task), so we can't draw definitive conclusions here.

5.3 Key Statistics

- **Average retries per episode**: 0.28 (spatial), 0.48 (object)
- **Most retries needed**: 3 attempts for drawer manipulation task
- **First-attempt success rate**: 72% (spatial), 60% (object)
- **Second-attempt recovery rate**: 83% of failures

5.4 Acknowledgments

This work was completed as a take-home assignment for PhD application to Penn State CSE under Prof. Huijuan Xu. All experiments were conducted using Google Colab with A100 GPU access, totaling approximately $20 in compute costs.

5.4.1 Use Of AI

Throughout the process of performing this assignment, I used AI tools such as Claude and Github Copilot to assist with code generation, debugging, and documentation. These tools were employed to enhance productivity and ensure code quality, while all core ideas, implementations, and analyses were developed independently. I also used AI to understand the codebases of the original papers and to help summarize key concepts, and also to do deep-research on the generalization problem in Embodied AI to identify resources and relevant papers(few mentioned in section 2.1).

References

[1] Kim, M. J., et al. (2024). *OpenVLA: An Open-Source Vision-Language-Action Model*. arXiv:2406.09246

[2] Shah, D., et al. (2024). *Learning Affordances at Inference-Time for Vision-Language-Action Models*. arXiv:2510.19752

[3] Zhao, T. Z., et al. (2023). *Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware*. Robotics: Science and Systems (RSS)

[4] Liu, B., et al. (2023). *LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning*. Neural Information Processing Systems (NeurIPS)

[5] Brohan, A., et al. (2023). *RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control*. Conference on Robot Learning (CoRL)

[6] Open X-Embodiment Collaboration (2024). *Open X-Embodiment: Robotic Learning Datasets and RT-X Models*. International Conference on Robotics and Automation (ICRA)

[7] Octo Model Team, Ghosh, D., Walke, H., Pertsch, K., Black, K., Mees, O., Dasari, S., Hejna, J., Xu, C., Luo, J., Kreiman, T., Tan, Y. L., Chen, L. Y., Sanketi, P., Vuong, Q., Xiao, T., Sadigh, D., Finn, C., and Levine, S. (2024). *Octo: An Open-Source Generalist Robot Policy*. Robotics: Science and Systems (RSS), Delft, Netherlands.

[8] Deshpande, A., Deng, Y., Ray, A., Salvador, J., Han, W., Duan, J., Zeng, K.-H., Zhu, Y., Krishna, R., and Hendrix, R. (2025). *GraspMolmo: Generalizable Task-Oriented Grasping via Large-Scale Synthetic Data Generation*. arXiv preprint arXiv:2505.13441.

[9] Zhai, X., Mustafa, B., Kolesnikov, A., and Beyer, L. (2023). *Sigmoid Loss for Language Image Pre-Training*. arXiv preprint arXiv:2303.15343.

[10] Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. (2024). *DINOv2: Learning Robust Visual Features without Supervision*. arXiv preprint arXiv:2304.07193.

[11] Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., Florence, P., Fu, C., Gonzalez Arenas, M., 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. (2023). *RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control*. arXiv preprint arXiv:2307.15818.

[12] Open-X Embodiment Collaboration (2023). *Open X-Embodiment: Robotic Learning Datasets and RT-X Models*. Dataset and project website. Available: https://robotics-transformer-x.github.io/.

[13] Liu, B., Zhu, Y., Gao, C., Feng, Y., Liu, Q., Zhu, Y., and Stone, P. (2023). *LIBERO: Benchmarking Knowledge Transfer for Lifelong Robot Learning*. arXiv preprint arXiv:2306.03310.

[14] Ma, Y., Song, Z., Zhuang, Y., Hao, J., and King, I. (2025). *A Survey on Vision-Language-Action Models for Embodied AI*. arXiv preprint arXiv:2405.14093.

[15] Firoozi, R., Tucker, J., Tian, S., Majumdar, A., Sun, J., Liu, W., Zhu, Y., Song, S., Kapoor, A., Hausman, K., Ichter, B., Driess, D., Wu, J., Lu, C., and Schwager, M. (2023). *Foundation Models in Robotics: Applications, Challenges, and the Future*. arXiv preprint arXiv:2312.07843.

[16] Hu, Y., et al. (2024). *Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis*. arXiv preprint arXiv:2312.08782.

[17] Zare, M., Kebria, P. M., Khosravi, A., and Nahavandi, S. (2023). *A Survey of Imitation Learning: Algorithms, Recent Developments, and Challenges*. arXiv preprint arXiv:2309.02473.

[18] Black, K., Galliker, M. Y., and Levine, S. (2025). *Real-Time Execution of Action Chunking Flow Policies*. arXiv preprint arXiv:2506.07339.

[19] Duan, Y., Yin, H., and Kragic, D. (2025). *Real-time Iteration Scheme for Diffusion Policy*. arXiv preprint arXiv:2508.05396.