Politecnico di Torino

Politecnico di Torino

Computer Engineering

Artificial Intelligence and Data Analytics

A.A. 2024/2025

Graduation Session December 2025

Adapting LLaMA 3.2 Vision for Unified Robotic Planning and Control

Relatori:

Flavio Esposito

Alessio Sacco

Guido Marchetto

Candidato:

Vittorio Di Giorgio

Abstract

Vision-Language-Action (VLA) models are emerging as powerful tools for embodied AI, allowing robots to merge visual perception with language understanding to execute complex tasks. Their potential lies in combining perception, reasoning, and control within a single framework, which could greatly enhance robotics pipelines and boost generalization.

However, it remains uncertain how effectively today's open, mid-size vision-language models (VLMs) can be adapted into practical VLAs. Previous works like RT-2 have demonstrated impressive results but rely on large proprietary models and undisclosed training methods, while OpenVLA presents an open-source alternative based on composite architectures. In contrast, this study investigates whether a single, open, mid-size model like LLaMA 3.2 Vision Instruct can be fine-tuned into a functional VLA using only limited computational resources, along with carefully designed prompting and training strategies.

The approach is evaluated on two complementary benchmarks: ALFRED, which emphasizes high-level household reasoning and long-horizon planning, and Open X-Embodiment (OpenX), which concentrates on low-level robotic manipulation trajectories. For ALFRED, the model is fine-tuned to generate both a natural language plan and a discrete sequence of actions (e.g., GoToLocation, PickupObject). This is achieved by creating structured prompts that enforce the chronological order of observations and conditionally integrate scene objects, effectively introducing dropout to enhance robustness. This framework enables the model to produce coherent and aligned language–action plans, even though the inference time per sample remains significant.

For OpenX, a discrete action vocabulary is established by mapping 256 uncommon LLaMA tokens to 8-dimensional robot actions (termination, 3D position, rotation, gripper). While single-frame baselines struggle to follow meaningful trajectories, reframing the task as multi-frame sequence prediction with temporal windows and object-focused prompts allows the model to maintain trajectory consistency, termination, and gripper control.

The results show that these fine-tuned models significantly surpass baseline configurations that lack fine-tuning or prompt design. Even when using only a small subset of the original datasets and training with limited resources, the models display promising abilities in both high-level reasoning and low-level control.

Overall, the findings suggest that mid-sized, open models like LLaMA 3.2 Vision can serve as effective foundations for embodied AI when combined with efficient fine-tuning and carefully engineered inputs. These insights illuminate future research directions, including scaling data and computational resources, refining

prompting strategies, and ultimately fostering the development of general-purpose, resource-efficient VLA systems for robotics.

Acknowledgements

First and foremost, I would like to express my sincere gratitude to Dr. Flavio Esposito, at Saint Louis University (SLU) in Missouri for welcoming me during this incredible international experience. Working on my thesis abroad has been an extremely formative and enriching opportunity that allowed me to grow both academically and personally, exposing me to a stimulating and diverse research environment beyond the Politecnico di Torino.

I am especially thankful to the Computer Networks Laboratory, for their constant support, valuable feedback, and guidance throughout the entire project. I also wish to express my appreciation to Dr. Madi Biabasil and researcher Guangping Liu from the Mechatronics Laboratory for their assistance, insightful discussions, and collaboration during my time in St. Louis.

I am deeply grateful to Dr. Alessio Sacco and Prof. Guido Marchetto for giving me the opportunity to carry out this project and for their continuous encouragement and availability.

This thesis would not have been possible without the support of all the above-mentioned individuals and institutions. Thank you for contributing to a unique and unforgettable academic journey.

Table of Contents

List of Figures

Chapter 1

Introduction

The ability to instruct a robot using natural human language, transforming verbal instructions into executable physical actions, has long been a core ambition in robotics and artificial intelligence. Traditionally, to perform even simple tasks, such as "to place the cup in the sink" requires a complex and fragile modular pipeline. These pipelines typically involve object detectors, pose estimators, symbolic planners, and motion controllers, each of which must be separately trained, manually integrated, and precisely calibrated. This fragmentation creates systems that are difficult to scale, fragile in edge cases, and highly dependent on domain-specific engineering.

In recent years, the emergence of large multimodal models has opened the door to a fundamentally different approach. These models jointly process language and vision, enabling them to reason over images and textual input in a unified architecture. One such model is LLaMA 3.2 Vision-Instruct, the first vision-language model in the LLaMA family, which combines a visual encoder and an 11-billion-parameter language decoder to perform grounded reasoning and instruction following. Using its capacity to interpret scenes and respond to natural language, this model offers a potential route to collapse the traditional pipeline into a single, learned system. More recent generations in the same family, such as LLaMA 4, further extend this line with mixture-of-experts architectures and very long context windows, but in this dissertation we intentionally focus on a medium-sized, widely accessible backbone as a realistic starting point for robotic applications.

This dissertation investigates whether such a model can be used to program a robotic arm, receiving natural language instructions and producing appropriate robot actions, without relying on explicitly engineered intermediate steps. We explore three progressively grounded output modalities:

- Natural-language task plans: textual sub-task lists describing what to do step-by-step,

- Symbolic plans: structured action representations encoded in JSON,
- Low-level control vectors: compact 8-token outputs encoding motion and manipulation commands.

The study is organised around four central research questions:

1. To what extent can a frozen LLaMA 3.2 Vision-Instruct model, conditioned only through prompting, generate coherent and plausible task plans on the ALFRED benchmark?
2. How much does parameter-efficient fine tuning with LoRA on ALFRED improve the model's ability to produce structured and executable plans, both in natural language and as discrete JSON action traces, and which failure modes remain?
3. Can the same backbone, equipped with dedicated LoRA adapters, be reused as a controller on the Open X-Embodiment dataset, mapping visual observations and instructions to 8-token control vectors that capture meaningful continuous behaviour?
4. Under realistic resource constraints, what trade offs arise when compressing these adapted models with 4 bit weight-only quantisation, in terms of memory and latency gains versus degradation of planning and control performance?

To answer these questions, we design a multi-phase experimental pipeline. We begin by evaluating the model's reasoning capabilities using one-shot and few-shot prompt engineering techniques, such as Chain-of-Thought (CoT), ReAct-style reasoning and visual-based prompting, in a curated subset of ALFRED tasks. We then perform supervised fine-tuning on the full ALFRED dataset to teach the model how to generate executable natural-language and symbolic action plans. In the next phase, we adapt the same architecture to low-level robotic control using Open X-Embodiment demonstrations, training it to predict discrete action vectors suitable for real-time manipulation. Finally, we apply quantization techniques to assess the feasibility of deploying the full pipeline on edge hardware.

The remainder of this dissertation details the methodology, datasets, and empirical results across each stage.

Chapter 2

Background and Related Work

Recent years have seen fast progress in vision-and-language models, and more broadly, in the use of large language models (LLMs) within robotics. These systems combine visual perception with natural-language reasoning and generation, allowing robots to interpret images, understand verbal instructions, and produce action descriptions or control commands in a single architecture. When scaled, LLMs provide rich semantic knowledge and strong capabilities for symbolic reasoning and long-horizon planning.

Yet, a central challenge remains: grounding these capabilities in the physical world. A text-only LLM has no direct access to sensor data or robot dynamics, so it cannot guarantee that its plans are feasible or safe when executed on a particular platform [1]. The physical environment does not respond to abstract descriptions but to concrete motor commands that must respect geometry, kinematics, and contact constraints.

To bridge this gap, recent work has proposed multimodal and hybrid architectures that fuse vision, language, and low-level control into unified models [2]. Vision-language-action systems learn from paired trajectories, images, and instructions so that perception, planning, and action are optimized jointly rather than engineered as separate modules. This marks a shift away from traditional modular pipelines toward foundation-style models that can reason about tasks while remaining connected to sensory input and motor output.

This chapter first reviews key previous work along this trajectory and then introduces the approach adopted in this dissertation: adapting LLaMA 3.2 Vision–Instruct to ALFRED and Open X-Embodiment, constructing task-specific datasets, and designing a training and evaluation pipeline that spans high-level planning and low-level control.

2.1 Classical Modular Pipelines in Robotics

Robots have traditionally followed a Sense-Plan–Act architecture: sensors capture data (e.g., cameras, LiDAR), which a perception module processes to estimate the environment. A planner generates actions or trajectories, and a controller executes them [2].

This modular design allows for clean separation between perception, planning, and control, making analysis and debugging easy to manage. The approach is backed by strong theoretical foundations in optimal control and motion planning.

However, this decomposition shows limitations in the real world. Small upstream errors cascade downstream, and handcrafted components require extensive per-task tuning. Pipelines designed for "pick-and-place" tasks often fail in more complex settings like "folding laundry."

To overcome these issues, end-to-end learning has emerged [3, 4]: models map raw sensory input directly to motor output, bypassing intermediate engineered representations. The idea is that, given sufficient data and compute, a network can learn the pixel-to-action mapping more effectively than handcrafted systems [5].

While the balance between engineered and learned components remains debated [2], the emergence of foundation models, large vision-language-action systems trained on diverse robot data, has opened new avenues. The next sections examine these models and their enabling datasets.

2.2 Advanced Vision-Language Models for Robotic Control

2.2.1 Gato: A Multimodal Generalist Agent

A key milestone in unifying robot-control policies is Gato, introduced by DeepMind in 2022 as a multimodal, multi-task generalist model [3]. Gato is a single autoregressive Transformer trained on over 600 tasks, including dialogue, Atari games, image captioning, and robotic manipulation. It can output text tokens, gamepad commands, or joint torques from the same weights, depending on context.

This is enabled by modality-specific embeddings for text, vision, and proprioception, all decoded as unified token sequences.

Although smaller than current LLMs (~1.18B parameters), Gato matched domain-specific systems on simple skills like block stacking [3]. Although it did not surpass handcrafted baselines, it demonstrated the feasibility of multitask, multimodal learning in a single Transformer, paving the way for robot-centric foundation models.

2.2.2 BC-Z: Generalisation from Natural Task Descriptions

BC-Z is a large-scale imitation learning framework designed for zero-shot generalization to new tasks using language or video prompts [6]. Instead of task IDs, BC-Z uses semantically rich inputs like text or videos of humans performing the task.

The architecture combines a text/video encoder with a visual convolutional backbone, feeding into a policy network for a 7-DoF arm. It was trained with behavior cloning on 25,877 demonstrations across 100 tasks via teleoperation and shared autonomy. The resulting policy generalizes zero-shot to unseen objects and task goals.

2.2.3 Robotics Transformer: RT-1 and RT-2

Google Robotics introduced RT-1 and RT-2 to combine Transformer flexibility with robot-specific data.

RT-1 is a decoder-only Transformer trained on over 130,000 teleoperated episodes spanning 700 domestic tasks [4]. It encodes instructions and image sequences via a visual backbone and the TokenLearner module, producing 11 discrete tokens representing a 7-DoF arm and mobile base. Despite its discrete action space, RT-1 behaves as a continuous policy, generalising across novel tasks, distractors, and long-horizon plans, outperforming earlier models like Gato and BC-Z.

RT-2 builds on RT-1 by co-training robot data with web-scale vision-language knowledge [5]. Leveraging large pretrained models such as PaLI-X and PaLM-E [7], RT-2 encodes robot actions as text tokens and learns from both real-world robot demonstrations and Internet-scale imagery and captions. This results in emergent abilities absent in robot-only training, for instance recognising unseen objects, interpreting symbolic references (e.g., “on square 5”), or reasoning over object attributes (e.g., “pick the smallest”).

An even greater generalization was achieved with the Open X-Embodiment dataset [8], comprising over 1 million demonstrations from 60 sources and 22 robot morphologies. Fine-tuning RT-1 and RT-2 on this dataset produces RT-1-X and RT-2-X, which significantly outperform their predecessors. RT-2-X, for example, triples success on unseen tasks and exhibits finer spatial understanding (e.g., distinguishing “on the table” from “near the table”). This confirms that scaling data diversity across robots, tasks, and environments enhances generalisation while reducing the need for task-specific fine-tuning.

However, RT-2 and its Open-X variants are not publicly released, and their architectures are tightly coupled to the PaLI-X and PaLM-E backbones. In this dissertation, we instead adopt LLaMA 3.2 Vision–Instruct as a medium-sized, publicly available backbone and take inspiration only from the high-level design of RT-style discrete action tokens, using an 8-token parameterization for termination,

Cartesian deltas, rotations, and gripper state while designing our own training and evaluation pipeline.

2.2.4 Language-Driven Planning: SayCan

A complementary line of work treats large language models (LLMs) as high-level planners while delegating physical execution to low-level controllers. SayCan [1] couples a 540B-parameter PaLM LLM with a mobile manipulator. The LLM acts as a semantic "brain," decomposing a user instruction into candidate sub-actions (e.g., navigate, grasp, open drawer). Each primitive skill is paired with a learned value function that estimates its chance of success in the current scene. At each step, the system selects the candidate that maximizes the product of what the LLM says and what the robot can do, hence the name "Say" + "Can".

This grounding-by-affordances mechanism enables the robot to take advantage of the rich world knowledge of the LLM while filtering out infeasible suggestions. Experiments show that the robot can execute multi-step kitchen tasks expressed in natural language, such as "Bring me a Coke from the living room; if none is present, fetch a bottle of water from the kitchen," requiring conditional planning and object reasoning.

The study underscores that, while an LLM can propose plausible plans in the abstract, linking its output to real-time perception and embodied value functions yields a far more reliable agent. SayCan therefore exemplifies a loosely coupled integration: the LLM plans, and the robot executes, constrained by affordance scores.

2.2.5 PaLM-E: An Embodied Multimodal Language Model

PaLM-E [7] pushes the integration of language and embodiment further by embedding a large-scale LLM (PaLM) inside an end-to-end multimodal architecture that directly ingests robot sensory data. RGB images, joint angles, and other state signals are encoded into PaLM-compatible "sentences" and interleaved with natural-language instructions. The entire sequence is processed by a unified Transformer, jointly fine-tuned on diverse embodied tasks, robotic planning, visual question answering, and image captioning, alongside web-scale text corpora.

At 562B parameters, PaLM-E achieves strong cross-domain generalization: it solves manipulation tasks across multiple robot platforms and achieves state-of-the-art results on knowledge-intensive benchmarks such as OK-VQA, all while maintaining its original linguistic capabilities.

Scale and task diversity prove mutually beneficial: vision-language pretraining enhances robotic reasoning, and exposure to physical-world tasks does not degrade, and can even improve, pure NLP or vision performance. PaLM-E thus demonstrates

the viability of a single foundation model that can understand, perceive, plan, and explain, an important milestone toward general-purpose robotic intelligence.

2.2.6 Large-Scale Vision-Language Models: Flamingo and PaLI-X

While not originally designed for robotics, large-scale vision-language models have laid key foundations by serving as visual backbones or pretraining components. Two prominent examples are Flamingo and PaLI-X.

Flamingo [9] is a family of few-shot multimodal models that integrate a pretrained vision encoder (e.g., EfficientNet or ViT) with a language model (e.g., GPT) via cross-modal attention layers. This allows flexible processing of image-text sequences. Without any task-specific fine-tuning, Flamingo achieves state-of-the-art results on open-ended tasks such as visual question answering, image captioning, and video understanding, using only a few in-context examples. Its web-scale multimodal training enables generalization to novel domains and has directly influenced robotic systems, such as RoboFlamingo [10], which uses Flamingo-like backbones for visual manipulation.

PaLI-X [11], derived from the PaLI (Pathways Language & Image) project, expands this paradigm with a multilingual encoder-decoder architecture. Trained on over 25 vision-language tasks, including captioning, document reading, object recognition, and video QA, PaLI-X sets state-of-the-art performance across nearly all benchmarks. It demonstrates emergent capabilities such as accurate object counting and multilingual grounding, even without explicit supervision.

These generalization trends mirror those in large monomodal LLMs and confirm the benefits of scale and task diversity. In robotics, models like RT-2 directly inherit PaLI-X's weights as vision-language encoders and policy initializers, bringing its broad perceptual and semantic knowledge into embodied settings.

More recently, the LLaMA family has introduced multimodal variants such as LLaMA 3.2 Vision–Instruct and, subsequently, LLaMA 4. These models pair visual encoders with large language backbones and, in the case of LLaMA 4, adopt a mixture-of-experts design in which only a subset of parameters is activated for each input, improving efficiency and scalability. Although they are not specifically tailored for robotics, their ability to process images and text jointly, together with extended context windows aimed at long-range reasoning, makes them attractive candidates for history-aware control policies. This dissertation explores this direction with LLaMA 3.2 Vision–Instruct as an accessible medium-sized backbone, while Chapter 10 discusses how future work could leverage newer LLaMA generations.

Chapter 3

LLaMA 3.2 Vision Instruct

3.1 The LLaMA 3.2 Vision Instruct Model

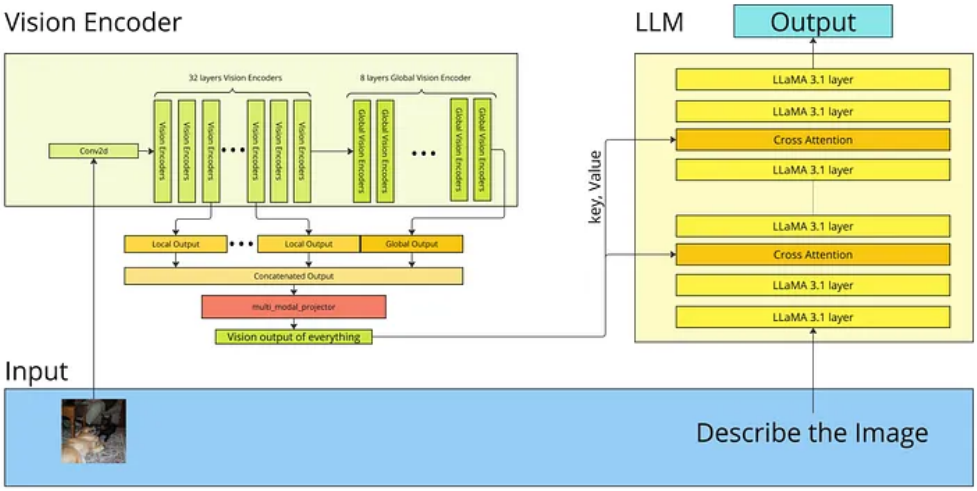


Figure 3.1: Schematic view of the LLaMA 3.2 Vision-Language architecture, with a dual-path vision encoder and cross-attention layers in the language decoder. Source: [12]

LLaMA 3.2 Vision Instruct is a multimodal extension of Meta's LLaMA 3.1 language model, designed to handle both text and image inputs within a single autoregressive framework. While the original LLaMA 3.1 model is purely text-based and comes in 11B and 90B parameter versions, the Vision variant adds a dedicated visual processing module that allows the model to reason about visual content in conjunction with language.

At the core of this extension is a modular vision adapter that connects a two-stage vision encoder to the language model. The encoder consists of a 32-layer local transformer followed by an 8-layer global transformer, with gating mechanisms that regulate the flow of visual information. Intermediate features from several layers (e.g., layers 3, 7, 15, 23, and 30) are preserved, projected into the language model's token space, and fed into the LLM via cross-attention layers inserted at regular intervals throughout the transformer stack.

This selective integration, sometimes described as controlled fusion, avoids both early and late fusion extremes and instead allows the model to attend to visual features precisely when and where they are needed during text generation. As a result, LLaMA 3.2 Vision is able to combine fine-grained visual perception with high-level reasoning in a way that remains fully compatible with the autoregressive nature of the language model.

The model accepts image and text as input and produces text as output. It supports up to 128k tokens of context, making it suitable for long-form, multi-turn interactions. While the language model itself is multilingual, English is currently the only officially supported language for vision-related tasks.

Thanks to its architecture, LLaMA 3.2 Vision performs well on a wide range of multimodal benchmarks, including image captioning, visual question answering, and visually grounded instruction following. Moreover, because the core language model can remain frozen during fine-tuning, the system is especially well-suited for domain-specific adaptation such as robotics where visual grounding is required but preserving the model's linguistic capabilities is essential.

LLaMA 3.2 Vision Instruct offers a well-balanced and flexible approach to multimodal reasoning. Its design reflects a thoughtful combination of modularity, efficiency, and scalability, making it a strong candidate for real-world applications that rely on both perception and language understanding [12].

3.2 Positioning within Language Model Architectures

Transformer-based language models can be broadly categorized into three main paradigms: Causal Language Models (CLMs), Masked Language Models (MLMs), and Sequence-to-Sequence (Seq2Seq) models, each defined by distinct training objectives and suited to different downstream applications.

LLaMA 3.2 belongs to the family of CLMs, which are trained with an autoregressive objective, predicting each token based solely on past tokens in the input sequence. This causal constraint ensures that the model processes text from left to

right, without access to future context during generation. Such models are widely adopted for open-ended generation and instruction-following tasks.

In contrast, Masked Language Models like BERT are trained to recover randomly masked tokens using full bidirectional attention. This enables them to capture rich contextual dependencies and makes them highly suitable for discriminative tasks such as classification or token-level inference. However, their design does not naturally support sequential generation without auxiliary mechanisms.

Sequence-to-Sequence models, such as T5 and BART, adopt an encoder-decoder structure. The input sequence is first processed by a full-context encoder, and then decoded token by token using an autoregressive decoder. This architecture is particularly suited to tasks that require structured input-output transformations, including summarization and translation. While these models combine bidirectional and autoregressive processing, they tend to be more complex in both architecture and training dynamics.

Understanding the distinctions between these modeling paradigms provides essential context for evaluating how LLaMA 3.2 Vision may be adapted to perform multimodal reasoning and robotic planning tasks, which are explored in the methodology presented in the following chapter.

Chapter 4

Methodology

This chapter presents the updated methodology adopted to adapt a pre-trained vision-language model to robotic manipulation tasks at multiple levels of abstraction. The training curriculum was refined to probe data efficiency, the impact of prompt conditioning, and the integration of temporal grounding through multi-frame inputs. A high-level overview of the pipeline is provided in the following sections.

4.1 Task Formulation

The project is structured around three complementary tasks:

- Natural language planning: generation of procedural instructions in free-form text, conditioned on visual sequences and task goals.
- Symbolic action planning: prediction of structured action sequences, encoded as JSON objects containing discrete high-level actions with arguments.
- Low-level action prediction: generation of 8-dimensional robot control vectors discretized into 8-token sequences (termination flag, 3D world vector, 3D rotation vector, and scalar gripper state).

Each capability was first addressed in isolation and then integrated into a shared fine-tuning and evaluation framework.

4.2 Prompt-Based Evaluation

Before fine-tuning, the frozen model (LLaMA 3.2 Vision Instruct) was evaluated on the full Test Unseen split of the ALFRED dataset (648 samples) using several prompting strategies, including zero-shot, one-shot, few-shot, chain-of-thought

(CoT), visual-focused, and RaR (Rephrase and Respond). These results established the baseline for subsequent fine-tuning.

4.3 Task-Specific Fine-Tuning

Fine-tuning was performed using Low-Rank Adaptation (LoRA), inserting low-rank adapters into both attention and MLP blocks while keeping the pretrained backbone frozen. Hyper-parameter details are reported in Chapter 6.

4.3.1 ALFRED Training

ALFRED fine-tuning was conducted in two main phases reflecting the progressive introduction of task structure and supervision.

Phase A: Natural-language planning (50% split, 15 images). The model was trained on a random 50% subset of ALFRED episodes using 15 RGB frames per sample and natural-language plans as targets. No auxiliary prompt, input dropout, or targeted conditioning was applied. This phase corresponds to the natural-language planner analysed in Section 7.5.

Phase B: Structured planning (full dataset, 15 images). The model was then fine-tuned on the full dataset to generate structured JSON outputs containing both the natural-language plan and the discrete action list required to complete each task, as detailed in Chapter 7. Earlier exploratory runs with intermediate data fractions (for example a 70% split with 10 frames) did not yield competitive results and are therefore omitted from the main analysis.

4.3.2 OpenX Robotic Control Training

The OpenX dataset provides paired visual observations and continuous robot control trajectories. Each timestep contains a termination flag, a 3D world vector, a 3D rotation vector, and a scalar gripper state. These four components form an 8-dimensional control vector that is discretized into an 8-token representation and mapped to a reserved range in the LLaMA tokenizer to ensure compatibility with the language modeling framework.

Two training regimes were explored:

Single-frame (baseline). Each sample included one RGB frame and the task instruction. The model predicted the subsequent 8-token action vector directly from this input.

Multi-frame temporal window. A second setup used a sliding window of four consecutive frames. The model predicted the next action conditioned on the visual history and previous actions. A custom prompt guided cross-attention toward relevant frames and spatial regions, improving temporal grounding without explicit motion supervision.

4.4 Evaluation Protocols

Evaluation is conducted separately for the ALFRED and OpenX domains, focusing on complementary dimensions of reasoning and control.

ALFRED. Evaluation distinguishes between two output formats. For the natural-language planning phases, text quality was measured using BLEU, ROUGE-1/2/L, and F1 overlap scores, complemented by an average composite score, format validity, and success rate. For the structured JSON planning phase, metrics assessed both the textual plan and the discrete action list, including JSON validity, action type, argument, and sequence similarity, and an overall comprehensive score aggregating plan quality and structural correctness.

Open X-Embodiment. Evaluation follows a multi-level scheme on the discretised 8-token action representation. Per-sample metrics include token-level, position, rotation, gripper, and termination accuracies, as well as exact-match ratio. Continuous and relative error variants are computed from detokenized vectors, reporting L2 and RMS-per-axis errors for position, rotation, and gripper components. Action-space analysis supplements these measures with directional accuracy, cosine similarity, and angular deviation between predicted and ground-truth action deltas. These scores summarize how well the model aligns its per-step commands with the reference commands over an episode, complementing discrete fidelity and continuous error with information about directional consistency and effective progress along the intended motion.

Quantized model evaluation. After fine-tuning, the best-performing models were quantized using the BitsAndBytes framework with a 4-bit NormalFloat (NF4) representation. This post-training quantization was applied directly to the LoRA-merged weights to assess how reduced precision affects model efficiency and inference speed. Quantized models were evaluated using the same metric suite adopted for the full-precision LoRA versions, allowing a direct comparison in terms of accuracy, latency, and computational cost.

Chapter 5

Datasets

5.1 Overview

This study relies on multimodal datasets that combine visual observations with natural language instructions and action level annotations. The selected corpora are designed to support learning across several levels of abstraction, ranging from high level planning in natural language to structured symbolic action sequences and low level robotic control vectors, so that the same backbone model can be evaluated consistently across tasks.

Two datasets are used primarily throughout this work. The first is the ALFRED dataset, a simulated environment featuring goal oriented household tasks annotated with natural language plans and symbolic action sequences. The second is the Open X-Embodiment dataset, a large scale collection of real world robotic demonstrations that spans a wide variety of platforms and behaviours.

Because of hardware constraints and the computational cost of multimodal training, only a fraction of the original data is utilised. Representative subsets are sampled from each dataset to balance coverage, diversity and tractability, while preserving the core structure and distribution of the full corpora and enabling practical experimentation on a single high memory GPU.

5.2 The ALFRED Dataset

The ALFRED (Action Learning From Realistic Environments and Directives) dataset [13] is a large scale benchmark designed to train and evaluate embodied agents in household environments. It combines visual observations, natural language instructions and structured action sequences within simulated indoor scenes rendered using the AI2 THOR simulator. Each episode provides a high level goal (for example “Put a hot cup in the microwave”) expressed in free form language,

accompanied by a corresponding demonstration trajectory composed of symbolic and low level actions.

Instruction: Put an alarm clock on the desk in the corner, above the drawers.

Ground truth plan:

1. Go straight to the small wooden desk with clocks on it.
2. Grab the alarm clock closest to the back wall and beside the lamp.
3. Turn to your right and walk to the end of the desk.
4. Place the alarm clock on the desk, above the drawers.

Figure 5.1: Example from the ALFRED dataset showing the 15 frame visual context, the natural language instruction and its corresponding four step reference plan.

What distinguishes ALFRED from other benchmarks is its emphasis on compositional generalization and temporally extended reasoning. Agents must perceive the visual scene, parse complex multi step instructions and execute structured sequences of manipulation and navigation steps in order to complete each task. The dataset therefore offers supervision at three distinct levels, namely high level natural language plans, intermediate symbolic action programs and low level primitive motor actions, which makes it particularly suitable for training and evaluating vision language action models across multiple layers of abstraction.

In this dissertation we focus on two supervision settings. The first is a text only regime for high level plan generation, while the second is a structured JSON format for instruction to action prediction that includes both free form language plans and sequences of discrete actions.

5.2.1 ALFRED Dataset Construction and Preparation

Natural language Planning Subset (text only)

ALFRED was adapted to the text only supervision setting by extracting the essential components required for high level plan generation. Each sample comprises:

- An instruction in natural language specifying the task goal.
- A sequence of 15 egocentric RGB frames uniformly sampled from the episode.
- A ground truth plan in natural language, detailing the sequence of subtasks.

For each trajectory multiple training samples are generated by using the different high level natural language descriptions (high_descs) provided in the annotation. These descriptions represent semantically equivalent alternative expressions of the same task, differing in phrasing and structure, and they increase linguistic diversity while preserving the underlying demonstration content.

Each training sample is constructed so that the instruction corresponds to the high level task description, while the ground truth response is obtained by concatenating the high_descs into a semicolon separated sequence that represents the full plan. Repeating this process across all annotation variants yields a dataset enriched with paraphrased examples suitable for robust plan generation.

Structured JSON Planning Subset

An alternative version of the ALFRED dataset was prepared using structured outputs in JSON format, retaining task specific structure by including a symbolic action sequence alongside the natural language plan. Each response is formatted as follows:

```
{
  "nl_plan": "Walk forward to the small desk in front of you;
              Pick up the middle alarm clock from the desk;
              Turn right and walk to the right side of the larger wooden desk;
              Put the clock down on the desk just right of the book.",
  "discrete_action_list": [
    {"action": "GotoLocation", "args": ["sidetable"]},
    {"action": "PickupObject", "args": ["alarmclock"]},
    {"action": "GotoLocation", "args": ["desk"]},
    {"action": "PutObject", "args": ["alarmclock", "desk"]},
    {"action": "NoOp", "args": []}
  ]
}
```

This format enables supervised learning of structured robotic plans with interpretable action steps. Discrete action sequences provide a compact and symbolic representation suitable for integration with downstream planning or control policies.

For each sample, a list of all detected objects in the scene is also stored:

```
"object_in_scene": [
  "CellPhone_3bc3970e", "Pen_2830e300", "Laptop_c51d9cc5",
  "AlarmClock_6316eacc", "Book_d16fbaa0", "CreditCard_a5a087c0",
  "TennisRacket_feb6b631"
]
```

This auxiliary information can be used to constrain predictions to available objects, to perform object conditioned decoding and to ground object mentions in a semantic scene context. The structured JSON format in turn supports precise evaluation by enabling token level comparison of action steps, argument correctness and task completion criteria.

Taken together, the natural language only and symbolic JSON variants support complementary training paradigms in which the same input, consisting of an instruction and a set of images, can be used either to train an instruction following planner or to train a task planner that predicts structured actions grounded in object and location semantics.

ALFRED Train and Test Partitions

Given the scale of the ALFRED dataset and the available computational budget, a reduced version was constructed for both training and evaluation. The subset maintains the structural and task diversity of the original benchmark while enabling tractable fine tuning and testing.

Specifically:

- Train split: 18,774 samples totaling 281,610 images, with each episode contributing 15 uniformly sampled RGB frames. The average instruction length is 9.24 words, and the average natural language plan is 79.25 words.

- Test Unseen split: 648 samples (9,720 images), representing novel tasks not encountered during training. Instructions average 9.30 words, with plans averaging 79.73 words.

Each split is balanced in the number of images per sample, which is always fifteen, and maintains consistent instruction styles. These subsets are used for all training and evaluation experiments in both text only and structured supervision settings.

5.3 The Open X-Embodiment Dataset

The Open X-Embodiment (OXE) dataset [14] represents a major milestone in robotic learning, as it aims to enable the development of generalist policies capable of operating across a wide range of robots, environments and manipulation skills. The corpus is constructed by merging sixty existing datasets collected from thirty four academic and industrial laboratories worldwide and spans twenty two distinct robotic embodiments, from single arm manipulators to bimanual platforms and quadrupeds. This large scale effort is motivated by the success of general purpose pretraining in areas such as natural language processing and computer vision and by the ambition to obtain comparable levels of generalization in robotics.

OXE provides more than one million real world trajectories stored in the RLDS (Reinforcement Learning Dataset Standard) format, which supports a wide range of sensor modalities including RGB, depth and point clouds, as well as diverse action representations. Data are organized hierarchically into episodes and steps: an episode corresponds to an entire execution of a task, from the initial observation to the final termination, while each step captures a single time instant and includes the observations (images, depth, robot state), the executed action, and associated metadata. The RLDS schema also allows efficient parallelized loading in major deep learning frameworks, making it a scalable choice for training transformer based policies that require substantial data.

Because the individual datasets contributing to Open X-Embodiment originate from different laboratories and robot platforms, their action formats are heterogeneous. Absolute end-effector poses are typically represented as 7D vectors (base_pose_tool_reached) combining 3D position and a 4D quaternion $(x, y, z, q_w, q_x, q_y, q_z)$, whereas incremental motions are often expressed as 6D deltas with three translational and three rotational components (for example Euler angles or axis–angle rotations). In this work we explicitly focus on incremental end-effector displacements rather than absolute poses: instead of predicting a 7D pose at each step, the model outputs 6D deltas (translation and rotation) that update the current pose over time. We therefore restrict attention to the RT-1 Robot Actions subset of Open X-Embodiment, where actions are already provided in this delta form, and operate throughout with this representation. This choice keeps the action space compact and easy to interpret while remaining expressive enough for the pick-and-place style manipulation considered here. Absolute end-effector poses are still present in the RLDS observations and can be reconstructed by integrating the deltas over time, but they are not used as inputs or training targets in our experiments.

Concretely, each continuous action in the RT-1 subset is composed of:

- a 2D base translation vector (base_displacement_vector $\in \mathbb{R}^2$) that specifies

the planar motion of the mobile base in the global frame;

- a scalar base yaw rotation (base_displacement_vertical_rotation $\in \mathbb{R}$) around the vertical axis;
- a 3D end-effector translation vector (world_vector $\in \mathbb{R}^3$) encoding Cartesian displacement of the tool;
- a 3D end-effector rotation delta (rotation_delta $\in \mathbb{R}^3$) encoding the change in orientation;
- a scalar gripper closedness value (gripper_closedness_action $\in \mathbb{R}$) encoding smooth gripper control;
- a three-way one-hot termination signal (terminate_episode $\in \{0,1\}^3$) indicating continue, terminate, or failure/timeout.

In this work we discard the two base-motion components and treat the base as fixed, concentrating instead on the 6D end-effector deltas (translation and rotation) plus gripper and termination. The resulting 8-dimensional control vector (6 for the arm, 1 for the gripper, 1 for the terminate flag) is the quantity that we tokenize and feed to the language model in later chapters.

The overarching goal of the dataset is to explore X embodiment learning, that is the ability to transfer knowledge across different robot morphologies, sensory inputs and manipulation tasks. Experimental results reported in [14] show that training high capacity models such as RT 1 [4] and RT 2 [5] on this merged corpus improves performance and generalization over single robot training baselines.

5.3.1 Open X-Embodiment Dataset for Robotic Control

Before distinguishing between single frame and multi frame variants it is useful to outline the rationale behind this two stage design. The initial goal was to test whether a backbone that was originally developed for language and general vision tasks, rather than for robotics, could be conditioned reliably to produce 8 discrete action tokens per step in the Open X-Embodiment setting. To keep this feasibility study tractable, a relatively small single frame subset was extracted from the original data and used as a baseline environment in which to validate the tokenization scheme, the prompting strategy and the loss formulation. Once this baseline confirmed that the method could learn meaningful mappings from images and instructions to 8 token actions, the focus shifted to a richer multi frame dataset, in which each sample carries temporal context and a larger number of episodes and trajectories are retained for training and evaluation.

Instruction: pick brown chip bag from top drawer and place on counter

Ground truth action (JSON):

```
{
  "base_displacement_vector": [0.0, 0.0],
  "base_displacement_vertical_rotation": [0.0],
  "gripper_closedness_action": [0.0],
  "rotation_delta": [0.0948, -0.0247, 0.1408],
  "terminate_episode": [0, 1, 0],
  "world_vector": [0.0417, 0.0015, -0.0655]
}
```

Figure 5.2: Example episode from Open X-Embodiment: egocentric visual observation, natural language instruction and corresponding ground truth action.

Action tokenization. Following the RT-1 codebase [4], continuous control vectors are converted into 8-token discrete actions before being fed to the language model. The termination flag is first obtained by applying an arg max over the three-dimensional one-hot vector, yielding a class index in {0,1,2} that becomes the first token and encodes whether the agent should continue, terminate the episode, or signal a failure/timeout. The remaining components (world_vector, rotation_delta, gripper_closedness_action) are discretized independently along each dimension: values are clipped to their nominal ranges ($[-1,1]$ for translations and gripper, $[-\frac{\pi}{2}, \frac{\pi}{2}]$ for rotations), linearly normalized to [0,1], and mapped uniformly to one of 256 bins via rounding. For convenience, the resulting integers in [0,255] are then shifted into a reserved vocabulary band (IDs 126000–126255), which is rarely used by natural text and therefore avoids collisions with human-readable tokens. At inference time, the reverse mapping is applied: discrete tokens are dequantized back to approximate continuous controls, allowing the model to operate entirely in token space while still interacting with real-valued robot actuators.
Formally, after tokenization each low-level action is represented as an 8-token vector

$$\text{Action} = \{t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7\},$$

where

- t_0 encodes the termination flag (three-way decision: continue, terminate, failure/timeout),
- $t_{1:3}$ encode the world position delta $(\Delta x, \Delta y, \Delta z)$,
- $t_{4:6}$ encode the rotation delta (r_x, r_y, r_z),
- t_7 represents the gripper closedness action.

Single Frame Open X-Embodiment Subset for Reactive Control

For the Open X-Embodiment dataset [14], training was conducted on the same subset used by the original RT 1 model [4], comprising 73,499 robot trajectories collected from a mobile manipulator in the Google Micro Kitchen environment. This subset was selected both to ensure comparability with prior work and to contain the computational cost of using the full dataset.

In the single frame configuration only one RGB frame per step is extracted and temporal history is discarded. Each training example consists of:

- One RGB image of the current step;
- One high level natural language instruction;
- One action represented as an 8-token string encoding termination, position, rotation and gripper state.

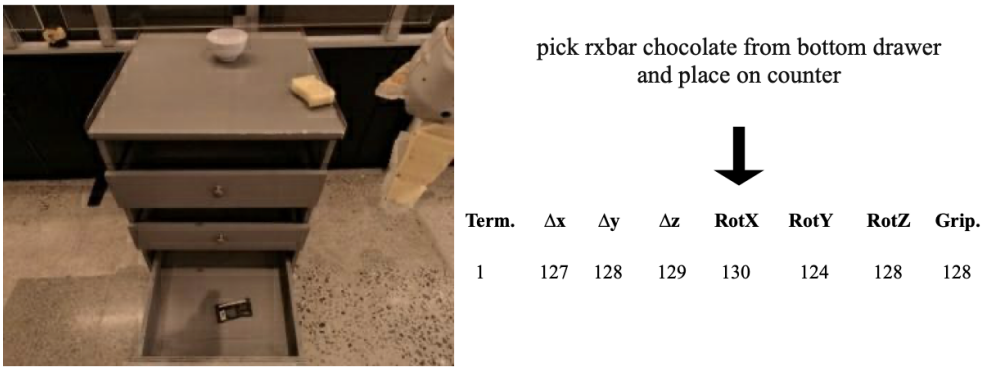


Figure 5.3: Example of instruction–action alignment in the Open X-Embodiment subset. The model receives a textual instruction (e.g., "move the gripper slightly upward and close") and produces an 8-token vector, where each token corresponds to a quantized control component (terminate flag, Cartesian displacement, rotation, and gripper state).

Dataset Statistics and Train/Test Split Table 5.1 summarises the main statistics of the single frame subset and its train/test partition. The split contains 48,625 samples overall, with 40,531 (83.4%) allocated to training and 8,094 (16.6%) to testing. These samples are drawn from 1,146 unique episodes. Across all splits, episodes comprise on average 42.43 timesteps (min: 2, max: 311); the training episodes have 2/42.84/311 steps (min/mean/max), while the test episodes have 2/40.47/122 steps. At the instruction level, the subset contains 316 distinct natural-language instructions overall, of which 307 appear in the training split and 66 in the test split.

Table 5.1: Single frame Open X-Embodiment dataset statistics.

Quantity	Value
Total samples	48,625
Training samples	40,531 (83.4%)
Test samples	8,094 (16.6%)
Unique episodes	1,146
Unique instructions (all/train/test)	316 (307 / 66)
Average steps per episode	42.43 (min: 2, max: 311)

Unless otherwise stated, all subsequent figures and tables describing ranges, means, and token histograms for the single frame subset are computed on the entire dataset prior to the train/test division. Separate inspections of the two splits yield empirically indistinguishable histograms, both in continuous and tokenised space, so the global distributions provide an accurate and non-redundant summary of the underlying data.

Continuous Action Statistics Although the action space is nominally defined over fixed ranges, translational deltas in $[-1,1]$ and rotational deltas in $[-\frac{\pi}{2}, \frac{\pi}{2}]$, the empirical distributions in Figure 5.4 show that, in practice, almost all actions occupy a much narrower region concentrated around zero. This indicates that the robot predominantly performs fine-grained adjustments rather than large displacements. The gripper signal exhibits a characteristic bimodal profile, reflecting the fact that it remains open for most of the trajectory and transitions to the closed state only when required to secure an object. These behavioral priors clarify why, in the evaluation discussed in Chapter 8, relative errors may appear substantial despite numerically small absolute deviations: the effective operating range is significantly compressed compared to the theoretical bounds, magnifying the perceived impact of small discrepancies.

Table 5.2: Ranges and means of continuous action components in the single frame setting.

Component	Min	Max	Mean
World Vector X	−0.6938	0.5551	0.0076
World Vector Y	−0.3443	0.3114	0.0065
World Vector Z	−0.6178	0.5100	−0.0105
Rotation Delta X	−1.1288	1.4320	0.0461
Rotation Delta Y	−0.9476	1.0538	−0.0059
Rotation Delta Z	−1.2851	1.3340	0.0039
Gripper Closedness	−1.0000	1.0000	0.0259

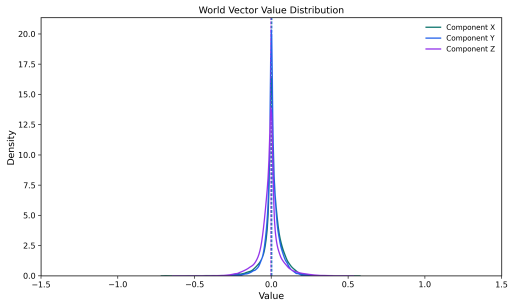
Values outside $[-1,1]$ for translations and $[-\frac{\pi}{2}, \frac{\pi}{2}]$ for rotations are clipped to the nearest boundary before tokenization, so they occupy the first or last discrete bin.

Discrete Token Statistics All continuous components were quantized into 256 discrete bins and mapped to reserved token IDs in the range from 126000 to 126255, producing an 8 token representation for every action.

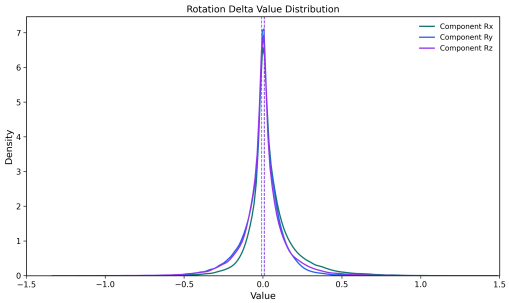
Table 5.3: Per component discrete token statistics in the single frame setting.

Component	Min Token	Max Token	Most Frequent Token (Count)
Terminate Episode	0	2	1 (46,194×)
World Vector X	39	198	128 (8,843×)
World Vector Y	84	167	128 (10,541×)
World Vector Z	49	193	128 (7,381×)
Rotation Delta X	36	244	128 (7,627×)
Rotation Delta Y	51	213	128 (7,593×)
Rotation Delta Z	23	236	128 (8,138×)
Gripper Closedness	0	255	128 (33,756×)

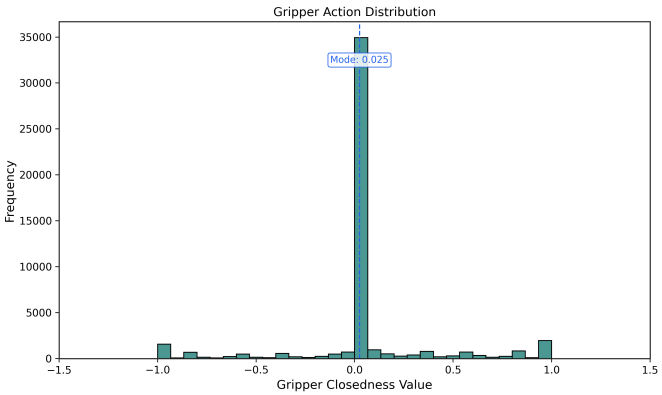
The bins span [0,255], but they are offset to 126000–126255 to avoid collisions with textual tokens, so ID 126000 corresponds to bin 0, ID 126001 to bin 1, and so forth. Because the continuous values cluster around zero, the central bin (token 128) accumulates most labels, leading to a strong imbalance: a naïve model can lower its loss simply by predicting the most frequent token. Chapter 8 therefore examines whether the learned policies genuinely use contextual signals to break this bias.



World Vector



Rotation Delta



Gripper Action

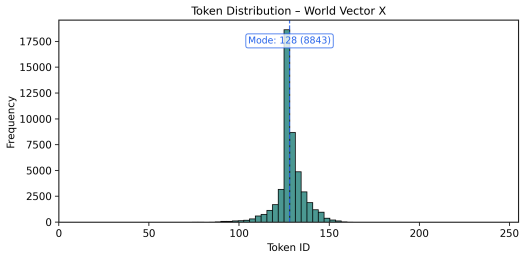
Figure 5.4: Continuous action distributions for the single frame Open X-Embodiment subset. Translation and rotation values remain concentrated near zero despite the nominal $[-1,1]$ and $[-\frac{\pi}{2}, \frac{\pi}{2}]$ ranges, while the gripper shows a bimodal pattern corresponding to mostly open or occasionally closed states.

Table 5.4: Terminate token class counts (discrete) in the single frame setting.

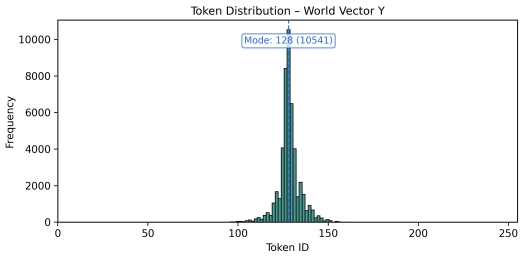
Split	Class 0 (Terminate)	Class 1 (Continue)	Class 2 (Failure flag)
All	2,290	46,194	141
Train	1,890	38,501	140
Test	400	7,693	1

The terminate statistics are instead reported separately for all, train, and test splits, as they quantify how many positive termination examples (class 0) are actually observed during training and highlight the scarcity of explicit failure flags (class 2) available to the model.

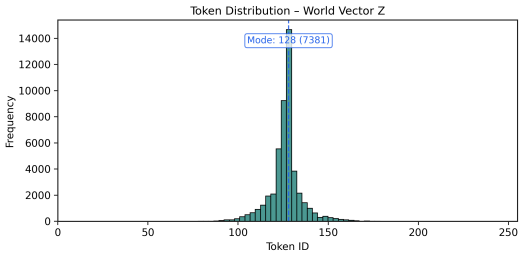
In the underlying continuous data the termination signal is encoded as a three dimensional vector. When an episode ends, a step with value [1,0,0] is followed by a final step with value [0,0,0]. During discretization both are mapped to the same terminate token (class 0), so every episode contributes two discrete termination labels even though only the first is explicitly one hot. Because the single frame subset includes a broader variety of tasks (not only pick style episodes), the failure flag (class 2) is more common here than in the multi frame split. Nevertheless, detecting failures is outside the scope of this work; the focus remains on predicting the continuous controls.



World Vector X

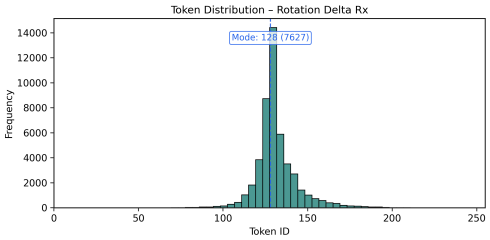


World Vector Y

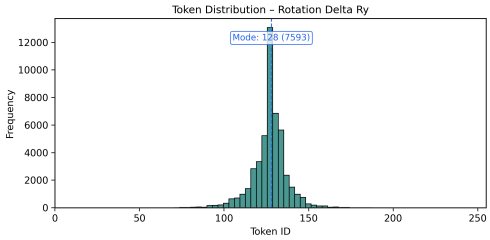


World Vector Z

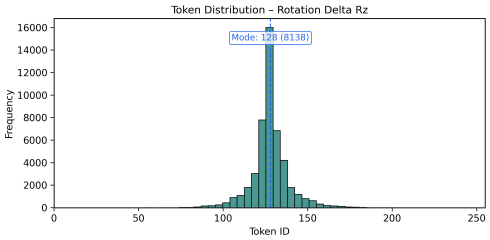
Figure 5.5: Token distributions for the three position components in the single frame dataset. All axes peak at the central bin (token 128), signalling that most samples correspond to near-zero displacements a key reason why priors dominate unless the model leverages visual context.



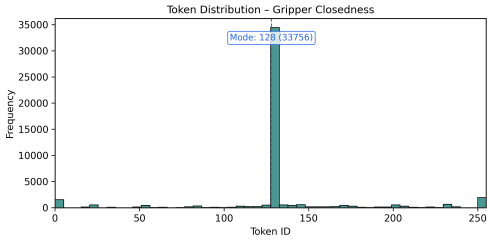
Rotation Delta Rx



Rotation Delta Ry



Rotation Delta Rz



Gripper Closedness

Figure 5.6: Token distributions for rotation components and gripper closedness in the single frame dataset. Rotation bins peak at token 128, indicating that most steps involve small angular corrections; the gripper exhibits a dominant central peak (steady open state) plus sparse counts near the extremes with 255 representing the full closed state and 0 the fully open state.

Multi Frame Open X-Embodiment Subset for Temporal Grounding

To improve temporal grounding and history aware control a multi frame version of the dataset was constructed, in which each training sample includes a sliding window of five consecutive egocentric RGB frames from the same episode. In this context, temporal grounding means that each predicted action is conditioned on a short history of observations and past actions, so that the policy reasons about an evolving trajectory rather than an isolated frame; Chapter 8 provides a more detailed discussion of grounding within the multi frame controller. This configuration captures the continuity of motion and objects over time. The Open X-Embodiment dataset [14] was used, selecting 2,300 episodes in total, with 2,000 used for training and 300 reserved for testing.

Each training example consists of:

- A window of four RGB images from consecutive timesteps;
- One high level natural language instruction;
- A list of three previous actions, each corresponding to the frame that precedes it in the window;
- An action represented as 8 discrete tokens that encode termination, position, rotation and gripper state.

Temporal alignment is ensured by matching each previous action to the corresponding preceding frame. When fewer than five steps are available, for instance at the start of an episode, black placeholder images are inserted and padding token ID 126256 is used for missing previous actions. Chapter 8 analyses how this temporal grounding influences the behaviour of the controller.

Dataset Statistics Table 5.5 summarises the multi frame partition and its train/test split. The subset contains 127,381 samples from 2,300 episodes, with 110,752 (86.9%) allocated to training and 16,629 (13.1%) to testing. Every sample includes four historical frames, the current frame, and three previous actions, and the 72 instruction types focus on pick style behaviours. Across all splits, episodes comprise on average 55.38 timesteps (min: 2, max: 200); the training episodes have 2/55.38/200 steps (min/mean/max), while the test episodes have 2/55.43/134 steps. At the instruction level there are 72 unique natural-language instructions overall, with 70 distinct forms in the training split and 55 in the test split.

Table 5.5: Multi frame Open X-Embodiment dataset statistics.

Quantity	Value
Total samples	127,381
Training samples	110,752 (86.9%)
Test samples	16,629 (13.1%)
Unique episodes	2,300
Unique instructions (all/train/test)	72 (70 / 55)
Average steps per episode	55.38 (min: 2, max: 200)

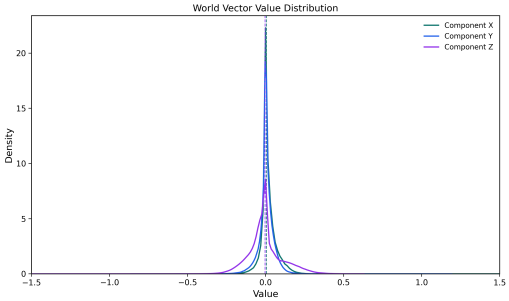
As for the single frame subset, all subsequent figures and tables reporting ranges, means, and token histograms for the multi frame data are computed on the entire multi frame dataset before the train/test split. The corresponding histograms for the two splits are nearly identical in both continuous and tokenized representations, so reporting global distributions is sufficient for characterizing the statistics of the control signals.

Continuous Action Statistics Also in the multi frame setting the effective operating range is much smaller than the nominal bounds: Figure 5.7 shows that position and rotation deltas remain sharply peaked around zero, with only slightly heavier tails than in the single frame case. The gripper continues to exhibit a bimodal pattern with a dominant open state. Consequently, the relative error metrics reported in Chapter 8 must be read in the context of this compressed range, as apparently large percentages often correspond to sub-centimetre deviations in absolute terms.

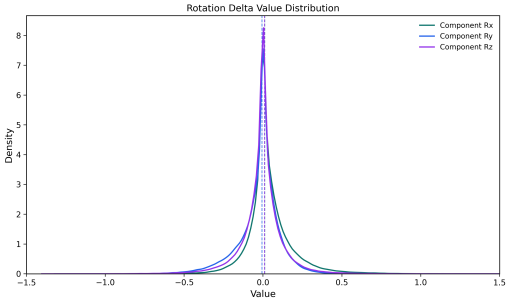
Table 5.6: Ranges and means of continuous action components in the multi frame setting.

Component	Min	Max	Mean
World Vector X	−0.8482	2.1878	0.0176
World Vector Y	−0.5564	1.1785	0.0043
World Vector Z	−1.7527	0.5993	−0.0021*
Rotation Delta X	−1.2811	1.5506	0.0385
Rotation Delta Y	−1.0808	0.9406	−0.0225
Rotation Delta Z	−1.3646	1.5675	−0.0074
Gripper Closedness	−1.0000	1.0000	0.0017

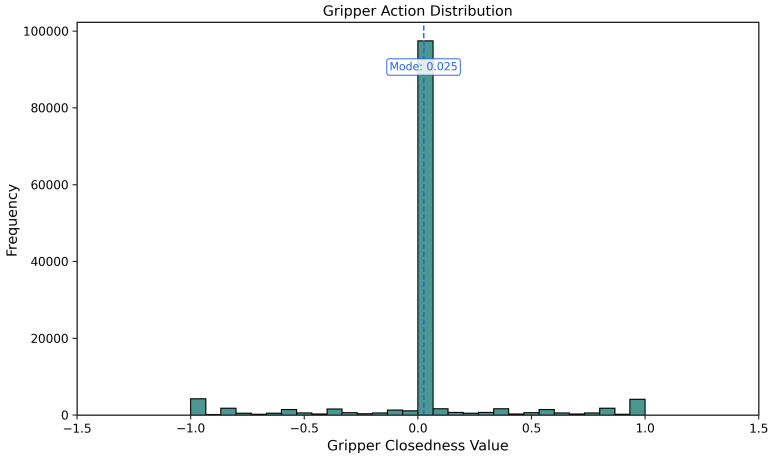
*Values outside the nominal ranges are saturated to the nearest boundary ([−1,1] for world vectors and gripper, $[-\frac{\pi}{2}, \frac{\pi}{2}]$ for rotations) before discretization, so they effectively collapse into the first or last bin during tokenization.



World Vector



Rotation Delta



Gripper Action

Figure 5.7: Continuous action distributions for the multi frame Open X-Embodiment subset. Translation and rotation retain a pronounced peak near zero with slightly heavier tails than in the single frame case, while the gripper remains mostly bimodal.

Discrete Token Statistics Quantization in the multi frame split follows the same scheme used for the single frame data: 256 uniformly spaced bins in [0,255] mapped to IDs 126000–126255, yielding an 8 token representation per action.

Table 5.7: Global token statistics.

Metric	Value
Most frequent token	ID 128 (83,879×)
Second most frequent token	ID 1 (46,194×)
Second most frequent positional token	ID 128 (10,541×)
Least frequent token	ID 12 (2×)

Table 5.8: Per component token statistics.

Component	Min Token	Max Token	Most Frequent Token (Count)
Terminate Episode	0	2	1 (122,758×)
World Vector X	19	255	128 (26,223×)
World Vector Y	57	255	128 (27,531×)
World Vector Z	0	204	128 (13,907×)
Rotation Delta X	24	253	128 (20,175×)
Rotation Delta Y	40	204	128 (20,401×)
Rotation Delta Z	17	255	128 (21,725×)
Gripper Closedness	0	255	128 (96,152×)

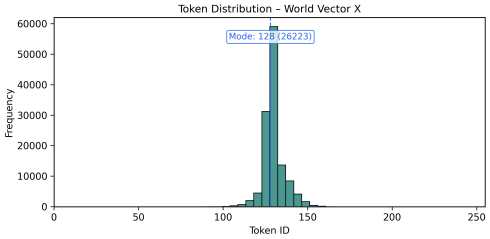
As before, the central bins dominate because most motion deltas lie near zero, and the gripper signal is mostly bimodal with an open state prevailing. Temporal context helps the model resist the temptation to predict the most frequent token blindly, but the imbalance still affects learning and motivates the relative error analysis in Chapter 8. The terminate token counts now reflect the fact that this split focuses on pick style episodes, which fail far less often than the broader single frame subset.

Table 5.9: Terminate token class counts (discrete) in the multi frame setting.

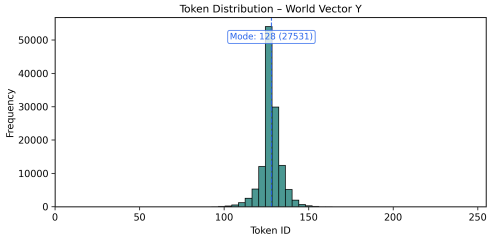
Split	Class 0 (Terminate)	Class 1 (Continue)	Class 2 (Failure flag)
All	4,600	122,758	23
Train	4,000	106,735	17
Test	600	16,023	6

Here too the terminate counts are stratified by split to make explicit how many terminating steps are available during optimisation and to emphasise that failure events are effectively negligible in the multi frame training data.

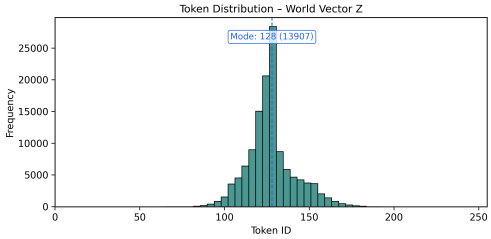
As in the single frame case, termination in the continuous trajectories is represented as a vector that switches to [1,0,0] and is then followed by a concluding step with [0,0,0]. Both of these are discretised to the terminate token, so each episode yields two discrete terminations near its end. Because the multi frame split focuses on pick tasks that rarely fail, the failure flag (class 2) is almost absent. As noted earlier, identifying failures is not a direct goal of this work; these statistics simply provide context for the termination metrics reported later.



World Vector X

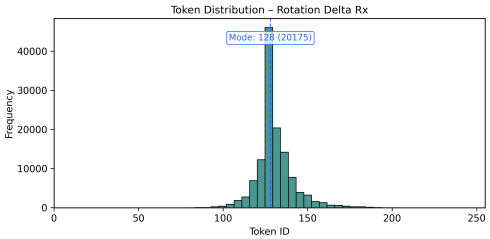


World Vector Y

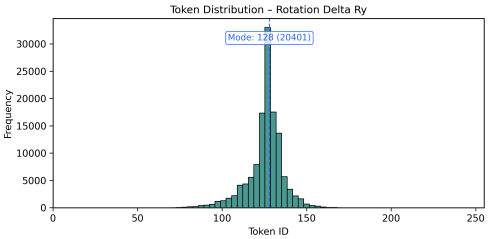


World Vector Z

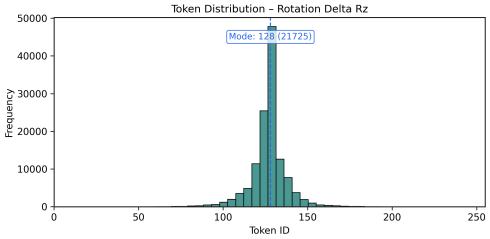
Figure 5.8: Token distributions for positional components in the multi frame dataset. The mass concentrates near the central bins, indicating small motions around a neutral pose. Termination counts are summarised separately.



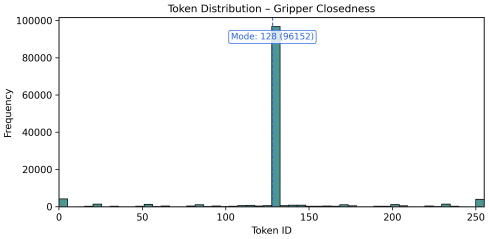
Rotation Delta RX



Rotation Delta RY



Rotation Delta RZ



Gripper Closedness

Figure 5.9: Token distributions for rotational components and gripper closedness in the multi frame dataset. Components are centred around the neutral bin, reflecting small rotational deltas and modest gripper adjustments.

Chapter 6

Experimental Setup

6.1 Hardware and Software Environment

All experiments were conducted on a single NVIDIA A100 GPU with 80GB of VRAM, a setup commonly used in high-performance deep learning research. The software stack included Python 3.10, PyTorch 2.3.0, HuggingFace Transformers 4.51.3, PEFT 0.10.0, and DeepSpeed 0.13.4. Training progress and evaluation metrics were monitored via TensorBoard.

Instantiating the base LLaMA 3.2 Vision–Instruct 11B checkpoint on this hardware already occupies roughly 23 183 MiB of GPU memory (about 23 GiB), before any task-specific heads or visual inputs are added. In later chapters we will see that the effective footprint during inference can grow substantially above this baseline, depending on the number of image frames, the length of the prompt, and whether quantization is applied.

To ensure reproducibility, random seeds were fixed for all relevant libraries, and all configuration files used in the experiments are archived in the project repository github.com/Vittorix99/VLARobotics.

6.2 Training Framework and Implementation

The training process was implemented using a modular framework built on top of HuggingFace Transformers and DeepSpeed, enabling scalable fine-tuning with support for parameter-efficient adaptation through the PEFT library.

Given the large size of the LLaMA 3.2 Vision-Instruct model (11B parameters), full fine-tuning was computationally infeasible. Instead, the model was fine-tuned using Low-Rank Adaptation (LoRA), updating only a small number of trainable matrices in the language and vision branches while keeping the backbone frozen. The image projection layer was also included in the optimization.

Mixed-precision training (FP16) and gradient checkpoint were activated to reduce memory consumption and increase training throughput.

Each training instance consisted of multimodal inputs and task-specific outputs:

- ALFRED: sequences of 15 RGB frames, paired with natural language instructions and either textual plans or symbolic JSON actions.
- Open X-Embodiment: single- or four-frame inputs, each paired with discretized 8-token vectors representing low-level robotic control actions.

All image inputs were resized to 560×560 pixels and normalized according to the vision encoder specifications. At this resolution, each image occupies a single visual tile out of the four supported by the LLaMA 3.2 Vision Instruct 11B architecture (maximum layout: 2×2 tiles for 1120×1120 inputs). Prompt templates followed an instruction-tuning format, while a custom tokenizer handled the mapping between symbolic or discretized actions and reserved token IDs, ensuring full compatibility with the multimodal LLM.

6.3 Training Configurations

The tables below summarize the core hyperparameters and settings used across all fine-tuning stages.

All reported experiments were carried out in FP16 precision. During subsequent troubleshooting, we observed that some long runs developed numerical instabilities in FP16 that did not appear when the same configuration was re-run in BF16. This behavior is consistent with the more limited dynamic range of FP16, whose five-bit exponent makes it more prone to overflow in intermediate quantities such as attention projections, normalization layers, or LoRA updates. Once infinities or NaNs appear in activations or gradients, they propagate through DeepSpeed's sharded parameters and dynamic loss scaling, eventually leading to training failures. In contrast, BF16 preserves the FP32 exponent range and proved numerically robust in our tests. Re-running all fine-tuning stages in BF16 was not feasible within the available budget, so the main results rely on FP16 together with gradient checkpointing to reduce memory usage by roughly one third. The cosine learning-rate schedule with a 0.03 warmup ratio was chosen to avoid abrupt changes during the first few hundred steps and to provide a smooth decay over the single training epoch, which is particularly important when fine-tuning LoRA adapters on heterogeneous multimodal data.

6.3.1 ALFRED Fine Tuning

Table 6.1: ALFRED fine tuning configurations used in the reported experiments.

Phase	Split	Images	Batch/Accum.	Prompting/Target
A	50%	15	2/2	Plan only (natural-language planning)
B	100%	15	1/4	Plan + JSON (structured planning, few shot prompt)

Phase A corresponds to the natural-language planning fine tuning analysed in Section 7.5 and uses a 50% subset of the ALFRED training data. Phase B extends the same backbone to the structured planner with JSON outputs described in Chapter 7 and is trained on the full training split. Several additional configurations with different data fractions or image counts were explored during early experimentation, but they did not provide consistent improvements and are therefore not discussed further.

Common hyperparameters.

- LoRA rank: 32; LoRA alpha: 32; LoRA dropout: 0.05
- Learning rate: 1×10^{-6}
- Projector learning rate: 5×10^{-5}
- Vision learning rate: 1×10^{-5}
- Weight decay: 0.1
- Optimizer: AdamW
- Precision: FP16
- Scheduler: cosine with warmup ratio 0.03
- Number of epochs: 1

Batch size and gradient accumulation were tuned empirically, gradually increasing the effective batch until the GPU reached its memory limit. A representative example is the ALFRED training run described in Chapter 7, where switching the accumulation factor from one to two halved the effective batch per step and led to a markedly smoother loss curve. The base learning rate of 10^{-6} was likewise selected after several trial runs, keeping the optimiser stable across tasks while leaving most of the adaptation burden to the LoRA layers. A rank of 32 with

matching alpha offered a good compromise between expressiveness and stability for an 11B-parameter backbone, in line with values commonly used for models in the 7B–13B range, without exhausting GPU memory when combined with 15-frame visual inputs.

6.3.2 Open X-Embodiment Fine Tuning

Table 6.2: Open X-Embodiment fine tuning configurations for the single frame and multi frame controllers.

Experiment	Frames	Batch/Accum.	Epochs	Base LR	Projector LR	Vision LR
Single frame	1	16/2	5	1×10^{-6}	1×10^{-5}	4×10^{-5}
Multi frame	5	4/4	1	3×10^{-6}	5×10^{-5}	1×10^{-5}

During preliminary experimentation we tested several learning rates, projector schedules and batch sizes, but the combinations in Table 6.2 offered the most stable convergence and were retained for all analyses. In particular, label smoothing was not used in the final Open X-Embodiment runs, as it led to unstable optimization under the Transformers version adopted in this work; the underlying issue and its impact on loss computation are documented in detail in Appendix A.

Other hyperparameters.

- Optimizer: AdamW
- Weight decay: 0.1
- LoRA rank: 32; LoRA alpha: 32; LoRA dropout: 0.05
- Precision: FP16
- Scheduler: cosine with warmup ratio 0.03

As for ALFRED, batch size and accumulation were selected by trial and error, probing how far the A100 GPU could be pushed without triggering out-of-memory errors. The higher visual and temporal load of the multi frame controller required a smaller per device batch than the single frame baseline, and the shared learning rate of 10^{-6} was validated through inspection of the loss curves. Projector and vision learning rates in the 10^{-5} range proved sufficient to adapt the visual pathway without destabilising the frozen backbone and, although FP16 precision could lead to numerical instability (a number of earlier FP16 runs had to be discarded due to the NaN-related instabilities discussed above), we have been able through a

Zero-2 DeepSpeed configuration and gradient checkpointing to complete the final single-frame (5 epochs) and multi-frame (1 epoch) Open X runs on a single GPU, albeit at the cost of several weeks of wall-clock time

6.4 Logging and Reproducibility

Training progress was tracked using TensorBoard, logging metrics such as total loss, token-level accuracy, and checkpoint snapshots every N steps. Random seeds were fixed across all libraries to ensure reproducibility. Complete configurations and scripts are available in the codebase.

LoRA adapters were crucial for enabling experimentation on a single A100 GPU while maintaining reasonable training speed and capacity. All results reported in the following chapters are obtained with label smoothing disabled, which empirically provided the most stable optimization behavior for the multimodal LLaMA backbone used in this dissertation.

Chapter 7

ALFRED Benchmark: Evaluation and Results

7.1 Overview of Phases A and B

This chapter evaluates the proposed Vision–Language–Action (VLA) planner on the ALFRED benchmark, tracing its evolution from purely linguistic planning to structured, executable task representations. We consider two complementary phases:

- **Phase 1 – Natural Language Planning (text-only):** The model generates step-by-step natural-language plans describing how to achieve a given goal from egocentric visual observations. Evaluation combines lexical metrics (BLEU, ROUGE, F1) and semantic alignment measures based on sentence embeddings to capture both surface and conceptual similarity to ground-truth plans.
- **Phase 2 – Structured Planning (text + discrete actions):** The model emits, alongside the natural-language plan, a corresponding sequence of symbolic actions (discrete action list). The structured representation is serialised as JSON so that downstream robotic controllers can parse and execute it without extra post-processing. This dual output keeps the planning interface readable while exposing precise procedures to the control stack.

Moving toward discrete actions strengthens the link between language and control: it surfaces explicit preconditions and arguments, enables safety checks before execution, and matches the symbolic interfaces required for embodied deployment. The remainder of the chapter combines quantitative metrics with qualitative inspection to understand how fine-tuning and output structuring improve grounding, coherence, and procedural reasoning on the TEST-UNSEEN split of ALFRED.

7.1.1 Task Definition and Output Format

Each episode provides a natural-language instruction and 15 egocentric frames sampled from an ALFRED trajectory. The planner must transform this input into phase-specific outputs:

Phase A (Natural-Language Planning). The model generates a semicolon-delimited sequence of imperative steps describing how to accomplish the task. This plan is purely textual and targets human readability while preserving the chronological order of sub-actions.

Phase B (Structured Planning with Discrete Actions). In addition to the Phase A plan, the model emits a discrete action trace encoded as JSON. Every entry specifies an action type together with its arguments (e.g., GotoLocation, PickupObject, PutObject), allowing downstream controllers to parse, validate, and execute the program without extra post-processing.

The dual-output format preserves compatibility with Phase A analyses while introducing a machine-interpretable specification for deployment.

7.1.2 Action Vocabulary

The discrete branch in Phase B relies on a fixed action vocabulary aligned with the ALFRED simulator. Each element of the structured trace consists of an action type and a list of arguments. Action primitives cover navigation (e.g., GotoLocation), manipulation (PickupObject, PutObject, OpenObject, etc.), and control markers such as NoOp. Arguments identify the semantic entities involved, such as objects, receptacles, or target locations. For example, the instruction “move the alarm clock from one desk to another” yields

```
{"discrete action list": [
  {"action": "GotoLocation", "args": ["sidetable"]},
  {"action": "PickupObject", "args": ["alarmclock"]},
  {"action": "GotoLocation", "args": ["desk"]},
  {"action": "PutObject", "args": ["alarmclock", "desk"]},
  {"action": "NoOp", "args": []}
]}
```

which enumerates the path (side table → desk) and the necessary object interactions.

7.2 Evaluation Metrics

7.2.1 Phase A: Lexical and Semantic Metrics

To assess natural-language plan quality we adopt a combination of lexical and semantic diagnostics that jointly measure linguistic accuracy, semantic grounding, and temporal consistency.

Lexical Metrics. Lexical metrics capture word- and phrase-level overlap between predicted and reference plans. They are commonly used in text generation and translation, providing a first approximation of fidelity and fluency.

- **BLEU** [15]: Sentence-level BLEU with smoothing to evaluate n-gram precision over short phrases.
- **ROUGE-1 / ROUGE-2 / ROUGE-L** [16]: F-measure variants computed over unigrams, bigrams, and the longest common subsequence.
- **Token F1**: Harmonic mean of token-level precision and recall, offering a robust indicator of coverage beyond exact lexical matches.

To consolidate these scores, we compute a weighted Average Score:

$$\begin{aligned}\text{Average Score} = {} & 0.10 \cdot \text{BLEU} + 0.30 \cdot \text{F1} + 0.10 \cdot \text{ROUGE-1} \\ & + \ 0.15 \cdot \text{ROUGE-2} + 0.35 \cdot \text{ROUGE-L}\end{aligned} \tag{7.1}$$

This weighting emphasises ROUGE-L (0.35) and Token F1 (0.30) to reward structural fidelity and overall lexical coverage. ROUGE-2 (0.15) keeps some local order sensitivity, while ROUGE-1 (0.10) and BLEU (0.10) provide a lightweight check on base vocabulary and historical n-gram precision without dominating the score.

Semantic Metrics Lexical overlap metrics such as BLEU [15] or ROUGE [16] fail to capture paraphrasing, reordered steps, or semantically equivalent reformulations. To evaluate deeper semantic alignment, both predicted and reference plans are embedded using the sentence-transformers/all-MiniLM-L6-v2 model [17, 18], and cosine similarities are computed between the resulting sentence embeddings. Each plan is segmented into atomic steps by splitting on semicolons, ensuring a one-to-one mapping between sub-actions.

Semantic Textual Similarity (STS) [19] The STS score measures the cosine similarity between the embeddings of the full predicted and reference plans:

$$\text{STS} = \cos\left(\mathbf{E}(P_{\text{GT}}), \mathbf{E}(P_{\text{Pred}})\right). \tag{7.2}$$

Values close to 1.0 indicate near-identical plans, scores in the range $[0.7, 0.9]$ suggest strong semantic overlap with minor differences, and scores below 0.4 typically correspond to unrelated tasks. Since STS operates on entire plan embeddings, it does not account for step count or order.

Stepwise Semantic Similarity (SSS) [20] Following the BERTScore formulation [20], SSS computes the mean cosine similarity between each ground-truth step and its best-matching predicted step via greedy one-to-one alignment:

$$\text{SSS} = \frac{1}{N} \sum_{i=1}^{N} \cos\left(\mathbf{e}(\text{GT}_i), \mathbf{e}(\text{Pred}_{i^\star})\right), \tag{7.3}$$

where $i^\star = \arg\max_j \cos\left(\mathbf{e}(\text{GT}_i), \mathbf{e}(\text{Pred}_j)\right)$. High SSS scores ($[0.7, 0.9]$) indicate strong step-level semantic alignment, while scores in $[0.4, 0.69]$ reflect partial matches, and values below 0.4 suggest weak or inconsistent correspondence. SSS evaluates step-level similarity irrespective of order.

Coverage@0.7 [21] Inspired by the thresholded coverage metric from Visual Programming [21], Coverage@0.7 quantifies the fraction of ground-truth steps with at least one predicted step exceeding a cosine similarity threshold of 0.7:

$$\text{Coverage@0.7} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\left\{ \max_j \cos\left(\mathbf{e}(\text{GT}_i), \mathbf{e}(\text{Pred}_j)\right) \geq 0.7 \right\}. \tag{7.4}$$

A value of 0.5 indicates that approximately half of the reference steps are semantically covered by the predicted plan.

Step Temporal Consistency (STC) [22] After identifying the best-matching predicted step index j_i for each ground-truth step i, STC computes the Pearson correlation between the ground-truth and matched predicted step indices:

$$\text{STC} = \text{corr}_{\text{Pearson}}\left(i, j_i\right) \in [-1, 1]. \tag{7.5}$$

High values ($[0.7, 1.0]$) indicate preserved temporal order, mid-range scores ($[0.3, 0.69]$) suggest partial consistency, and negative or low values ($[-1, 0.29]$) reflect disordered or reversed sequences.

Average Off-Diagonal Similarity (AvgOffDiag) [23] Extending the redundancy analysis from Inner Monologue [23], AvgOffDiag computes the mean cosine similarity of non-diagonal entries in the step-to-step similarity matrix:

$$\text{AvgOffDiag} = \frac{1}{|\Omega|} \sum_{(i,j)\in\Omega} \cos\left(\mathbf{e}(\text{GT}_i), \mathbf{e}(\text{Pred}_j)\right), \quad \Omega = \{(i,j) : i \neq j\}. \tag{7.6}$$

Values above 0.4 suggest redundant or overlapping steps, while values below 0.2 indicate a clean one-to-one correspondence.

Following Reimers and Gurevych [17], Ethayarajh [24], and Gao et al. [25], cosine similarity between L2-normalized sentence embeddings serves as a proxy for semantic relatedness. With all-MiniLM-L6-v2, values above 0.9 typically indicate near-paraphrases, $[0.7, 0.9]$ denotes strong conceptual overlap, $[0.4, 0.69]$ reflects moderate topical relatedness, and values below 0.4 suggest weak or no semantic alignment. In the ALFRED dataset, stepwise scores around $[0.4, 0.55]$ indicate partial alignment, where predicted plans capture the correct scenario but may merge or reorder actions. Unlike BERT, which exhibits anisotropy that can inflate cosine similarities [24], all-MiniLM-L6-v2 is contrastively trained to produce a more isotropic embedding space [25]. Thus, cosine similarities (e.g., $[0.4, 0.55]$) reliably reflect partial correspondence without requiring post-processing like whitening [26].

Auxiliary Diagnostics. Complementary indicators quantify structural correctness and computational efficiency:

- **Format Score**: In Phase A this checks the natural-language plan structure (semicolon-separated steps earn 1.0, single-clause plans 0.5, malformed strings 0.0); in Phase B it also verifies JSON compliance of the discrete action list, yielding 1.0 only when both branches are well-formed.
- **Inference Time**: Average generation latency per sample, useful for deployment evaluation.
- **Token Count**: Average total tokens across prompt and response, reflecting verbosity and memory usage.

The selected metrics provide several points of view to assess the planning quality, spanning lexical accuracy, semantic grounding, temporal structure, and computational cost, enabling fair comparison between zero-shot and fine-tuned models on ALFRED.

7.2.2 Phase B: Discrete Action Metrics and Procedural Diagnostics

Evaluation protocols in embodied AI can be broadly categorized into online and offline regimes. Online metrics (Success Rate, Goal-Condition Success, SPL) require executing predicted plans inside a simulator. Our study adopts an offline setup: plans are not executed in ALFRED, so we focus on metrics that directly compare the predicted natural-language plan and discrete action trace to ground truth, emphasizing fidelity and procedural structure.

For the discrete branch we report:

- JSON validation score (format_score$_{\text{JSON}}$), taking values in $\{0, 0.5, 1\}$ and defined as the average of two binary checks: whether the JSON output contains a field `nl_plan` whose value is a string, and whether it contains a field `discrete_action_list` whose value is a list.

- Action type similarity: Jaccard overlap[1] between the sets of action types T_G and T_P; we denote it by S_{type} and compute

$$S_{\text{type}} = \frac{|T_G \cap T_P|}{|T_G \cup T_P|}. \tag{7.7}$$

- Action sequence similarity: positional agreement between action types; we denote it by S_{seq} and compute

$$S_{\text{seq}} = \frac{1}{\max(m,n)} \sum_{i=1}^{\min(m,n)} \mathbb{1}\left[t_i^G = t_i^P\right]. \tag{7.8}$$

- Action argument similarity: Jaccard overlap between the aggregated argument sets U_G and U_P; denoted by S_{arg} and defined as

$$S_{\text{arg}} = \frac{|U_G \cap U_P|}{|U_G \cup U_P|}. \tag{7.9}$$

- Combined action score: arithmetic mean of the three components; we denote it by S_{comb} and compute

$$S_{\text{comb}} = \frac{1}{3}\Big(S_{\text{type}} + S_{\text{seq}} + S_{\text{arg}}\Big). \tag{7.10}$$

These metrics jointly assess both the textual plan and the discrete action trace, quantifying the fidelity of the predicted format, the semantic and structural overlap of action types, and the alignment of action arguments. Together, they provide a measure of plan accuracy and procedural coherence in the offline setting.

Procedural diagnostics. To expose interpretable failure modes we introduce a suite of diagnostics on the discrete action trace. They quantify what entities are manipulated (objects), where manipulations occur (containers and locations), and how frequently arguments appear. We partition argument coverage into

[1]Also known as the Jaccard index, it is defined for two sets A and B as $\frac{|A \cap B|}{|A \cup B|}$ and quantifies the proportion of shared elements.

three set-based F1 scores, one for each semantic category, and a global frequency-aware micro-F1 that reflects overall argument agreement across all categories. Let $A_{\text{GT}} = \{a_1^{\text{GT}}, \ldots, a_m^{\text{GT}}\}$ and $A_{\text{Pred}} = \{a_1^{\text{Pred}}, \ldots, a_n^{\text{Pred}}\}$ denote the ground-truth and predicted action sequences, and let $\mathcal{A}_{\text{GT}}^{\text{obj}}$, $\mathcal{A}_{\text{GT}}^{\text{cont}}$, and $\mathcal{A}_{\text{GT}}^{\text{loc}}$ denote the sets of object, container, and location arguments extracted from A_{GT}, with analogous definitions $\mathcal{A}_{\text{Pred}}^{\text{obj}}$, $\mathcal{A}_{\text{Pred}}^{\text{cont}}$, and $\mathcal{A}_{\text{Pred}}^{\text{loc}}$ for the predicted trace.

Object Coverage F1 (F_1^{obj}) Measures set-based alignment of object arguments. Define precision and recall as

$$P^{\text{obj}} = \frac{|\mathcal{A}_{\text{Pred}}^{\text{obj}} \cap \mathcal{A}_{\text{GT}}^{\text{obj}}|}{|\mathcal{A}_{\text{Pred}}^{\text{obj}}|}, \quad R^{\text{obj}} = \frac{|\mathcal{A}_{\text{Pred}}^{\text{obj}} \cap \mathcal{A}_{\text{GT}}^{\text{obj}}|}{|\mathcal{A}_{\text{GT}}^{\text{obj}}|}, \tag{7.11}$$

and compute

$$F_1^{\text{obj}} = \frac{2P^{\text{obj}}R^{\text{obj}}}{P^{\text{obj}} + R^{\text{obj}}}. \tag{7.12}$$

By convention, $F_1^{\text{obj}} = 1.0$ if both sets are empty, and $F_1^{\text{obj}} = 0.0$ if exactly one set is empty. This metric captures what is manipulated, ignoring multiplicity and order.

Container Coverage F1 (F_1^{cont}) Measures set-based alignment of container arguments; definitions mirror those above.

Location Coverage F1 (F_1^{loc}) Measures location agreement, highlighting navigation accuracy irrespective of action order.

Argument Micro-F1 (Type Bucket) Aggregates objects, containers, and locations into a frequency-aware micro-F1 that weights repeated arguments and reflects overall argument recovery. Let N_{TP} be the number of correctly predicted argument occurrences, N_{Pred} the total number of predicted arguments, and N_{GT} the total number of ground-truth arguments; we compute

$$F_1^{\text{micro}} = \frac{2N_{\text{TP}}}{N_{\text{Pred}} + N_{\text{GT}}}. \tag{7.13}$$

Argument Consistency Index (ACI) Weighted combination of the four coverage scores capturing consistent recovery of objects, containers, locations, and their joint distribution.

Canonical Pair Coverage Fraction of ground-truth canonical object-location pairs (mug, sink) recovered in the predicted trace.

Movement Precision and Recall ($P_{\text{move}}, R_{\text{move}}$) Precision/recall over destinations visited by GotoLocation actions, diagnosing navigation fidelity.

Extra and Missing Steps Extra steps are counted as $\Delta_{\text{extra}} = \max\{0, |A_{\text{Pred}}| - |A_{\text{GT}}|\}$ and missing steps as $\Delta_{\text{miss}} = \max\{0, |A_{\text{GT}}| - |A_{\text{Pred}}|\}$, averaged across the dataset to quantify over- and under-generation.

All action-based scores lie in $[0, 1]$: values above 0.8 indicate near-perfect overlap between predicted and reference traces; the 0.6–0.8 band signals strong agreement with occasional deviations, whereas 0.3–0.5 reflects partial alignment (often caused by merged or reordered actions).

This section presents the zero-shot evaluation of the LLaMA 3.2 Vision-Instruct model on the ALFRED dataset, focusing exclusively on high-level planning: generating natural language action plans from visual observations and task instructions. This evaluation investigates the model's innate ability to understand egocentric scenes and follow instructions, without any task-specific fine-tuning.

7.3 Zero-Shot Baseline Evaluation

This section presents the zero-shot evaluation of the LLaMA 3.2 Vision-Instruct model on the ALFRED dataset, focusing exclusively on high-level planning: generating natural language action plans from visual observations and task instructions. This evaluation investigates the model's innate ability to understand egocentric scenes and follow instructions, without any task-specific fine-tuning.

Task Definition: Natural Language Planning

Given a natural language instruction and a sequence of 15 egocentric images from an ALFRED episode, the model is prompted to generate a detailed multi-step natural language plan in the phase A or a JSON structure text containing both the natural language plan and the corresponding discrete list of actions describing the subtasks required to complete the goal. In both cases, the output should ideally follow a structured format (e.g., a semicolon-separated list of short imperative sentences or a well-formed JSON object containing nl plan and discrete action list).

Prompting Strategies

To guide the model in producing consistent and actionable plans, six prompt templates were designed and tested. For each prompt, we crafted two variants: one for Phase A (natural language plan only) and one for Phase B (natural language plan + discrete action list in JSON). Each prompt incorporates the same core elements, namely task instruction, visual context, and output format, but varies in style and structure to explore different elicitation strategies. The six prompting strategies are:

1. **One shot**: A single worked example of a natural-language plan, followed by the query episode; it primarily teaches the expected semicolon-separated format and the level of granularity for individual steps.

2. **Few shot**: Multiple demonstrations of related tasks (e.g., moving different objects between receptacles) to promote in-context generalization of both structure and semantics beyond a single scenario.

3. **Style**: A description written in the style of a robot instruction manual, encouraging concise, imperative commands with an emphasis on clear, low-level actions rather than free-form narration.

4. **RAR**: A three-stage template (Retrieve, Reason, Respond) that first asks the model to list relevant objects, locations, and actions, then to reason about start and goal states, and finally to produce the step-by-step plan.

5. **CoT**: A chain-of-thought template that explicitly instructs the model to analyze the scene and think through the logical sequence of actions before emitting the final plan, thereby scaffolding intermediate reasoning.

6. **Visual focus**: Explicit references to the visual frames that instruct the model to carefully inspect objects, positions, state changes, and spatial relationships in each image, enhancing grounding and encouraging each step to be tied to observable evidence.

These prompting strategies were selected to cover complementary dimensions of reasoning and control: from minimal examples (One-shot) to structured reasoning chains (CoT) and visually grounded descriptions (Visual focus). This diversity enables a systematic comparison of how different forms of contextualization affect linguistic fluency, semantic alignment, and temporal coherence in subsequent evaluations, and anticipates the central role that visual grounding will play in our analysis of stepwise semantic alignment.

Each template concludes with the common question

> "Given the images above, what are the sequential subtasks I should perform to successfully complete the task: '{instruction}'?"

which re-states the user goal while anchoring the model to the accompanying frames.

Output normalization. Prompted generations often begin with a preamble rather than the plan itself (e.g., "Based on the provided images, here is a step-by-step plan to complete the task 'task':"). In practice, these preambles consistently terminate with a colon followed by a newline (":\n"). Our cleaning rule removes

everything up to and including this marker, so that evaluation only considers the plan content. For step segmentation, we adopt the dataset convention that each step ends with a semicolon; the plan is therefore tokenized by splitting on ";". This guarantees that both lexical and semantic analyses operate on the intended step list in natural language.

7.4 Zero-Shot Baseline Evaluation (Phase A)

Zero-shot experiments follow the same offline protocol across both phases: we evaluate on the TEST-UNSEEN split (648 episodes) without simulator roll-outs, parse each model response, normalize it, and compute the metrics defined above.

7.4.1 Quantitative and Qualitative Results

We benchmark six prompting strategies on the ALFRED TEST-UNSEEN split (648 episodes, 9 720 egocentric frames). Tables 7.1 and 7.2 summarize the numerical scores (mean ± std), while Figures 7.1, 7.6 and 7.5 visualize the main trends (ranking by average score; best-prompt lexical vs semantic radars; stepwise vs plan-level similarity distributions).

Table 7.1: Phase 1 lexical metrics per prompt (mean ± standard deviation).

Prompt	Avg. Score	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	#Token	Inf. Time (s)
Few-shot	**0.341** ± 0.10	**0.093** ± 0.08	**0.415** ± 0.13	**0.225** ± 0.10	**0.357** ± 0.11	**59.199** ± 23.77	**16.835** ± 5.18
One-shot	0.333 ± 0.09	0.087 ± 0.08	0.403 ± 0.12	0.218 ± 0.09	0.347 ± 0.11	60.606 ± 32.46	17.750 ± 7.61
Visual focus	0.313 ± 0.09	0.070 ± 0.07	0.386 ± 0.12	0.205 ± 0.09	0.323 ± 0.09	71.830 ± 51.47	20.668 ± 11.02
Style	0.301 ± 0.08	0.076 ± 0.07	0.386 ± 0.10	0.189 ± 0.08	0.316 ± 0.08	68.715 ± 34.97	20.531 ± 8.03
Cot	0.303 ± 0.08	0.068 ± 0.07	0.375 ± 0.11	0.196 ± 0.09	0.314 ± 0.09	62.394 ± 29.14	19.183 ± 6.82
Rar	0.280 ± 0.10	0.070 ± 0.06	0.350 ± 0.14	0.173 ± 0.09	0.286 ± 0.11	138.201 ± 77.04	40.528 ± 20.49

Lexical metric analysis. Few-shot is the strongest prompt on textual metrics: weighted average (0.341), ROUGE-1 (0.415), ROUGE-2 (0.225), and ROUGE-L (0.357) outperform the alternatives (Tab. 7.1; Fig. 7.6, left). In this setting:

- **BLEU** is typically low (~0.09 here) because it relies on strict n-gram matching, heavily penalising paraphrases ("pick up" vs "grab"), word-order changes, and ignoring semantic equivalences ("put" vs "place").
- **ROUGE-1** measures unigram overlap: scores around 0.35–0.45 are good in open-ended NL, indicating that a substantial share of key words (objects, verbs, locations) match.

- **ROUGE-2** evaluates bigrams and local cohesion: it is lower than ROUGE-1 (~0.17–0.22; 0.225 for Few-shot), meaning the lexicon is correct but phrasing/order often changes.

- **ROUGE-L** (LCS) reflects global structure: scores around 0.30–0.40 indicate main actions appear in the right order, despite formulation variations or minor omissions.

In ALFRED, higher ROUGE suggests the model talks about the same scenario, but does not guarantee identical procedural structure.

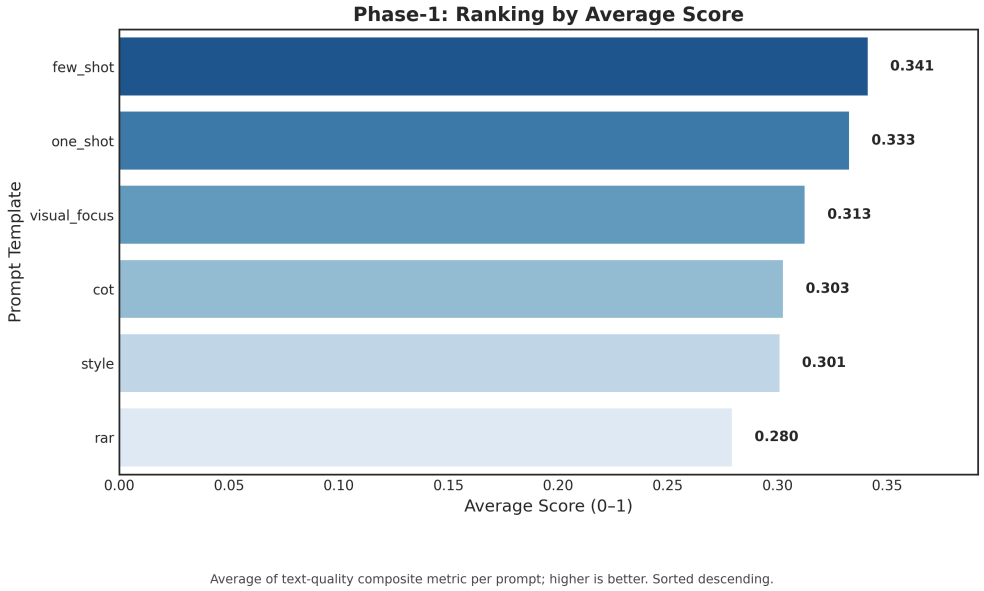


Figure 7.1: Ranking by weighted average score across prompts. Few-shot ranks first; bars also indicate dispersion across episodes.

Table 7.2: Phase 1 semantic metrics per prompt (mean ± standard deviation).

Prompt	STS	SSS	AvgOffDiag	Coverage@0.7	STC
Few-shot	0.756 ± 0.14	0.452 ± 0.11	0.443 ± 0.08	$\mathbf{0.516} \pm 0.22$	0.311 ± 0.39
One-shot	0.758 ± 0.13	0.445 ± 0.11	0.442 ± 0.08	0.509 ± 0.21	0.284 ± 0.40
Visual focus	$\mathbf{0.770} \pm 0.11$	$\mathbf{0.494} \pm 0.11$	$\mathbf{0.503} \pm 0.08$	0.468 ± 0.21	0.302 ± 0.40
Style	$\mathbf{0.770} \pm 0.09$	0.483 ± 0.09	0.485 ± 0.07	0.488 ± 0.21	0.273 ± 0.39
Cot	0.764 ± 0.09	0.485 ± 0.10	0.485 ± 0.07	0.483 ± 0.19	0.241 ± 0.38
Rar	0.700 ± 0.14	0.394 ± 0.13	0.429 ± 0.11	0.437 ± 0.21	0.269 ± 0.39

Semantical metric analysis. Across prompts, plan-level similarity is uniformly high: STS is ∼0.70–0.77, so predictions generally describe the same scenario as the references. Stepwise alignment is only moderate: SSS lies ∼0.39–0.49, with Visual focus near the top (≈0.49) and RAR at the bottom (≈0.39). Off-diagonal similarity is high (AvgOffDiag ∼0.44–0.50), signaling entanglement (redundant/overlapping steps). Few-shot and One-shot achieve the highest Coverage@0.7 (≈0.52/0.51) but do not substantially improve stepwise alignment or ordering; STC remains low (∼0.24–0.31) and variable. Overall, the typical profile is high STS, medium SSS, low STC. Visual focus tends to deliver the best stepwise compromise thanks to stronger visual grounding, while Few-shot/One-shot 'touch' more reference steps without ensuring consistent step-by-step or temporal fidelity.

SSS and STC capture complementary aspects: SSS measures semantic alignment at the step level, while STC measures whether those aligned steps appear in the correct temporal order. High SSS with low (or even negative) STC implies the right actions expressed in the wrong sequence. This is particularly evident when safety refusals or hallucinations disrupt the structure: in such cases, STC can approach zero or become negative (cf. Fig. 7.2). Two representative cases follow.

Limitations of prompt-only semantics. Even when prompt-only generation looks effective, several limitations remain for household task execution: (i) the output is not fully controllable, and safety or refusal behaviors can surface unexpectedly (cf. Fig. 7.2); (ii) the visual context is sampled into a short clip (14–15 egocentric frames), which may omit critical transitions. As a consequence, predictions can achieve high plan-level similarity (STS) while failing to preserve the correct temporal structure (low STC), as in Fig. 7.4. In practical terms, this means that a plan may "sound right" overall but still miss or reorder the precise steps needed to complete the task safely and efficiently in a real home environment. These observations motivate structured outputs and tighter control mechanisms beyond prompting alone.

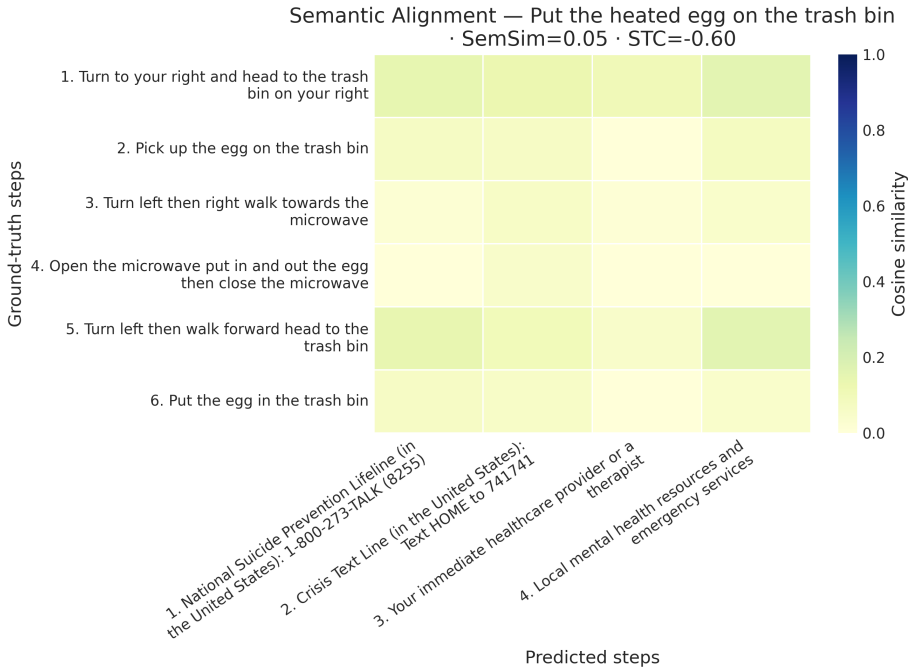


Figure 7.2: One-shot prompt, instruction "Put the heated egg on the trash bin". The generation triggered safety mechanisms (model refused the action and produced a helpline preamble).

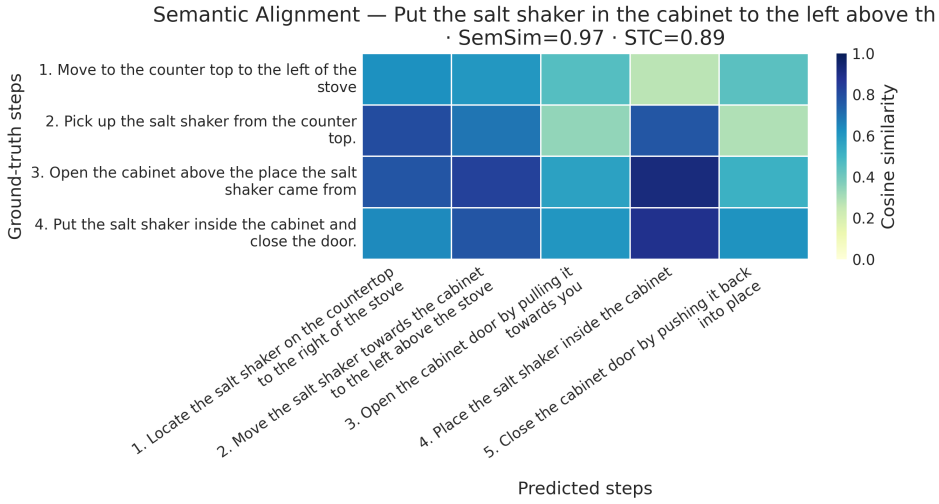


Figure 7.3: Semantic alignment heatmap (Visual focus, pick-and-place). Cells encode cosine similarity between ground-truth and predicted steps. The strong diagonal indicates high stepwise alignment (SSS), with ordering consistency reflected in a high STC; an extra final step appears off-diagonal.

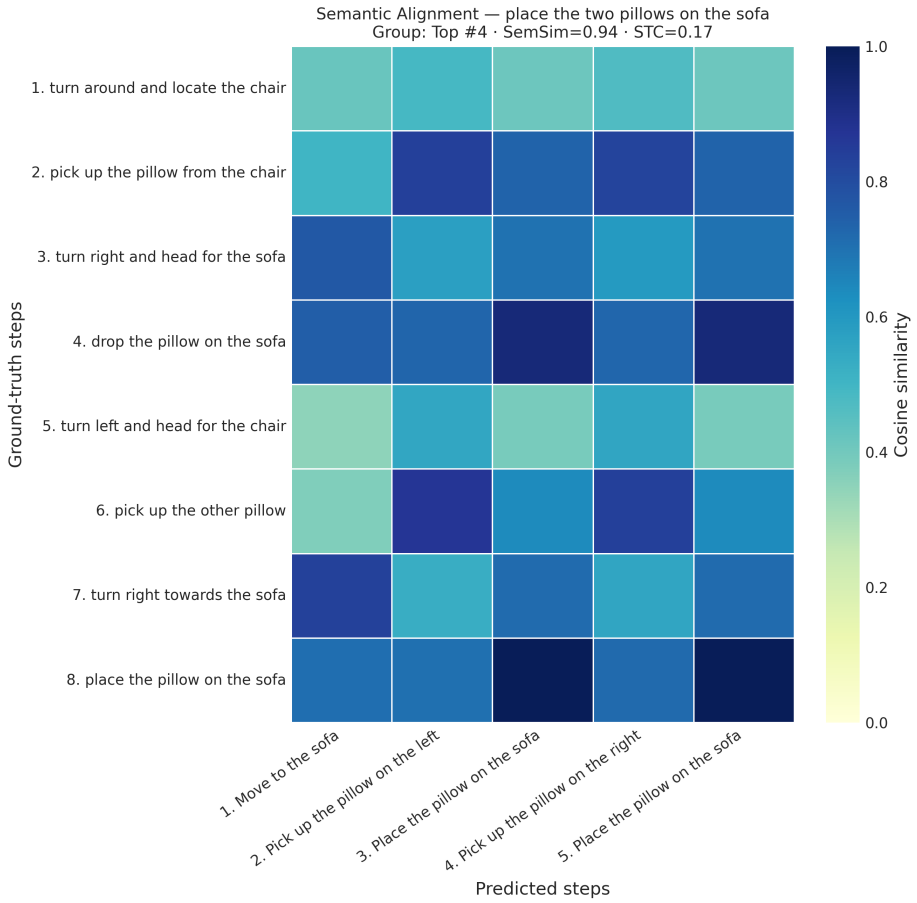


Figure 7.4: High STS but low STC (Visual focus, place two pillows). The plan-level meaning (STS=0.94) is close to the reference, yet the ordering is inconsistent (STC=0.17): diagonally weak bands and shifted peaks indicate reordering/merging of steps.

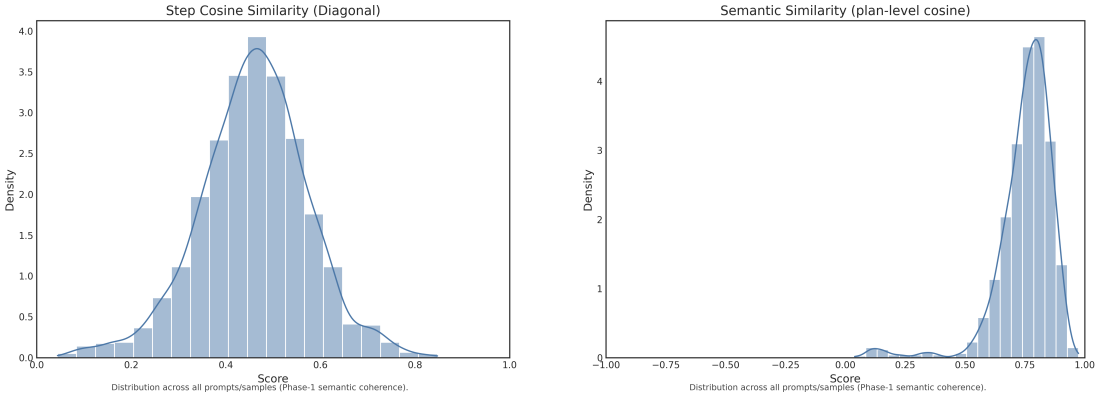


Figure 7.5: SSS vs STS distributions. Left: SSS measures step-by-step semantic similarity and is sensitive to inversions, missing steps, and merges; Right: STS measures plan-level similarity and is less sensitive to step order.

Figure 7.6: Side-by-side best-prompt radars. Left: lexical radar where BLEU is close to zero due to strict n-gram matching. Right: semantic radar where SSS is the lowest among semantic metrics, reflecting the difficulty of stepwise alignment.

Efficiency and verbosity. From a computational-cost perspective (Fig. 7.1), Few-shot achieves the lowest inference time (~16.8 s) with concise plans (~59 tokens). RAR is notably slower (~40.5 s) and verbose (~138 tokens), consistent with its three phase design (Retrieve to select relevant objects, Reason to compare initial and target state, and Respond to produce the final plan).

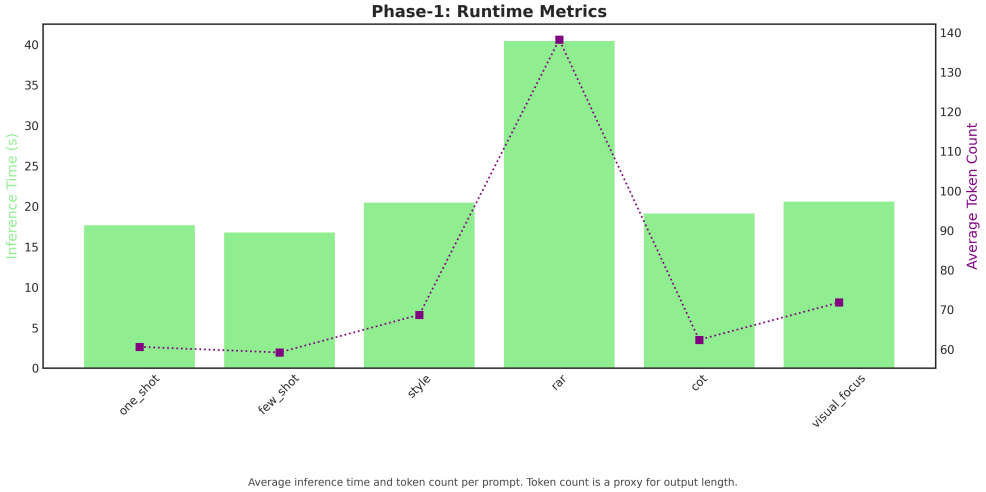


Figure 7.7: Average inference time (bars, left axis) and average token count (dashed line, right axis) per prompt. RAR incurs the highest latency and verbosity; Few-shot is the most efficient.

Qualitative comparisons. Beyond aggregate scores, we present full predictions across prompts in Tables 7.2, 7.3 and 7.4. The examples highlight systematic differences: few-shot and one-shot prompts, which include real in-context examples, tend to produce outputs whose structure more closely mirrors the reference plans; in contrast, stylistic or reasoning-oriented prompts can be more verbose or prone to local reorderings, despite comparable plan-level similarity.

Instruction	Move a knife in a cup to the sink.
Ground Truth	Turn right and move to stand in front of the sink. Pick up the butter knife from inside the sink. Turn left and move to stand in front of the coffee maker right after the stove. Open the cupboard above the coffee maker and place the knife in the green cup on the right. Pick up the cup with the knife inside from the cupboard. Turn left and carry the cup and knife to the sink. Place the cup with the knife inside in the sink.
One-Shot Output	Walk to the counter; Open the drawer; Take the knife in the cup from the drawer; Close the drawer; Walk to the sink; Place the knife in the cup in the sink.
Few-Shot Output	Walk to the counter; Open the cabinet door; Take a knife from the cabinet; Take a cup from the cabinet; Close the cabinet door; Walk to the sink; Place the knife in the cup in the sink.
CoT Output	1. Locate the knife in the cup; 2. Move the cup with the knife towards the sink; 3. Position the cup with the knife over the sink; 4. Place the cup with the knife into the sink.
Style Output	1. Locate the knife in the cup on the countertop; 2. Grasp the cup containing the knife; 3. Lift the cup off the countertop; 4. Carry the cup towards the sink; 5. Lower the cup into the sink; 6. Release the cup.
Visual Focus Output	1. Take the knife from the counter; 2. Place the knife in the cup; 3. Carry the cup with the knife to the sink; 4. Empty the cup into the sink; 5. Place the cup on the counter; 6. Put the knife in the sink.

Table 7.3: Qualitative comparison of model outputs for the same ALFRED task under different prompting strategies. Even prompts with high automatic scores diverge from the ground truth: typical issues include hallucinated objects (One-Shot, Few-Shot), oversimplification (CoT), excessive verbosity (Style), and grounding errors (Visual Focus).

Instruction	Pick up the white pencil on the desk and put it on another spot on the desk.
Ground Truth	Go to the right side of the wooden desk in front of you. Pick up the white pencil on the desk. Turn around and look behind you and then turn back around. Place the white pencil on the desk near the books.
One-Shot Output	Walk to the desk; Look for the white pencil on the desk; Pick up the white pencil; Walk to another spot on the desk; Place the white pencil on the other spot on the desk.
Few-Shot Output	Walk to the desk; Look for the white pencil on the desk; Pick up the white pencil; Walk to the other spot on the desk; Place the white pencil on the other spot on the desk.
CoT Output	1. Locate the white pencil on the desk; 2. Identify a suitable spot on the desk to place the pencil; 3. Pick up the pencil; 4. Move towards the identified spot; 5. Place the pencil on the new spot.
Visual Focus Output	1. Locate the white pencil on the desk; 2. Pick up the pencil; 3. Move towards the empty space on the desk; 4. Place the pencil on the empty space.
Style Output	1. Locate the white pencil on the desk; 2. Reach out to pick it up; 3. Grasp the pencil firmly; 4. Lift it off the desk; 5. Move to another spot; 6. Place it on the desk.
RAR Output	**Step 1: RETRIEVE** — Objects: White pencil on the desk, Desk, Bookshelf. **Step 2: REASON** — Initial state: pencil on desk. Goal state: pencil relocated. **Step 3: RESPOND** — 1. Pick up the pencil; 2. Move it; 3. Place it in a new spot.

Table 7.4: Qualitative comparison for a simpler relocation task. Here One-Shot and Few-Shot are almost perfectly aligned with the ground truth; CoT and Style add verbosity but remain valid; RAR follows a retrieve–reason–respond template.

7.5 Fine-Tuning on ALFRED – Natural-Language Planning (Phase A)

Fine-tuning in Phase A targets the main failure modes observed in the zero-shot prompts: plans sounded fluent but often missed, reordered, or duplicated actions. We therefore train on a 50% subset of our ALFRED dataset using 15 egocentric frames per episode, aiming to improve step alignment, temporal consistency, and redundancy control while keeping inference cost manageable.

Evaluation reuses the metric suite in Section 7.2. Lexical overlap (BLEU, ROUGE, F1) is complemented with semantic alignment diagnostics (STS, SSS, Coverage@0.7, AvgOffDiag, STC) and efficiency indicators, providing a holistic view of how supervision moves the model toward execution-faithful plans.

7.5.1 Training Configuration and Loss Trends

The fine-tuning setup follows the experimental framework in Chapter 6, using LoRA adaptation on LLaMA 3.2 Vision-Instruct 11B for one epoch with FP16 precision.

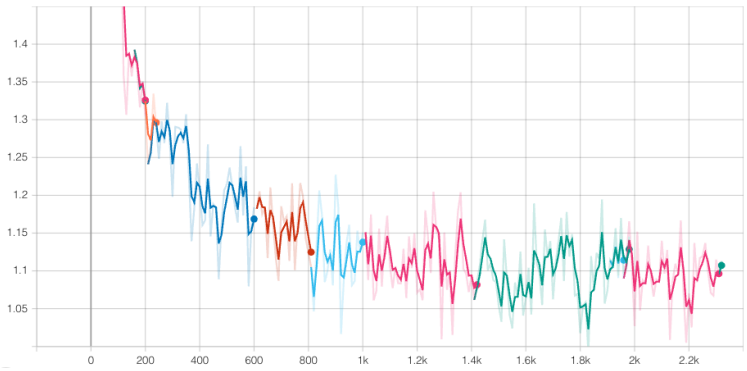


Figure 7.8: Training and validation loss curves for Phase A fine-tuning. The model demonstrates smooth convergence and stable optimization under LoRA adaptation.

The loss curve in Figure 7.8 shows a consistent and monotonic decrease up to approximately 1.1 across all resumed sessions, each represented by a different color segment corresponding to checkpoint restarts. The stability of this trend indicates smooth convergence and effective gradient flow under the LoRA configuration, with no signs of divergence or overfitting. Even limited supervised fine-tuning therefore proves sufficient to enhance the model's ability to produce coherent and temporally structured natural-language plans compared to purely prompt-based baselines.

7.5.2 Quantitative and Qualitative Results

We benchmark the fine-tuned model on the same ALFRED TEST-UNSEEN split introduced above (648 episodes, 9 720 egocentric frames). Table 7.5 reports the aggregate lexical metrics, while Table 7.6 focuses on semantic alignment.

Table 7.5: Phase A lexical metrics (fine-tuned model, mean ± standard deviation).

Metric	Mean ± Std Dev
Average Score	0.454 ± 0.094
BLEU	0.140 ± 0.093
ROUGE-1	0.547 ± 0.117
ROUGE-2	0.342 ± 0.101
ROUGE-L	0.463 ± 0.109
Token Count	47.91 ± 34.03
Inference Time (s)	13.98 ± 7.78

Table 7.6: Phase A semantic metrics (fine-tuned model, mean ± standard deviation).

Metric	Mean ± Std Dev
STS	0.845 ± 0.069
SSS	0.664 ± 0.104
AvgOffDiag	0.525 ± 0.058
Coverage@0.7	0.730 ± 0.184
STC	0.474 ± 0.352

Lexical metric analysis. Fine-tuning closes much of the lexical gap left by prompt-only generation. The composite score climbs from 0.341 for the Few-shot prompting (Tab. 7.1) to 0.454, with the highest gains on ROUGE-2 (+0.12 absolute) and BLEU (doubling to 0.14). This points to better local sequencing and richer n-gram overlap, while ROUGE-L reaching 0.463 confirms that the global action order is now tracked more faithfully. The fine-tuned model is also more economical: average responses shrink to 48 tokens and inference drops to 14 s, outperforming Few-shot prompting (59 tokens, 16.8 s). Figure 7.9 (lexical radar) summarizes these outward shifts.

Semantic metric analysis. Plan-level alignment benefits in parallel. STS rises to 0.845 and Coverage@0.7 to 0.73, indicating that most ground-truth steps now

find a high-similarity counterpart compared with the prompt-only spectrum in Tab. 7.2. Stepwise alignment (SSS) grows to 0.664 and AvgOffDiag falls, signaling reduced step entanglement, while STC nearly doubles relative to the best prompt. However, the large variance on STC reveals residual inversions on harder episodes. The distributions in Fig. 7.10 highlight how the tail of poorly aligned plans shrinks yet persists.

Figure 7.9: Phase A radar plots. Left: lexical metrics expand uniformly after fine-tuning. Right: semantic radar shows higher SSS and STC, though temporal consistency remains the weakest axis.

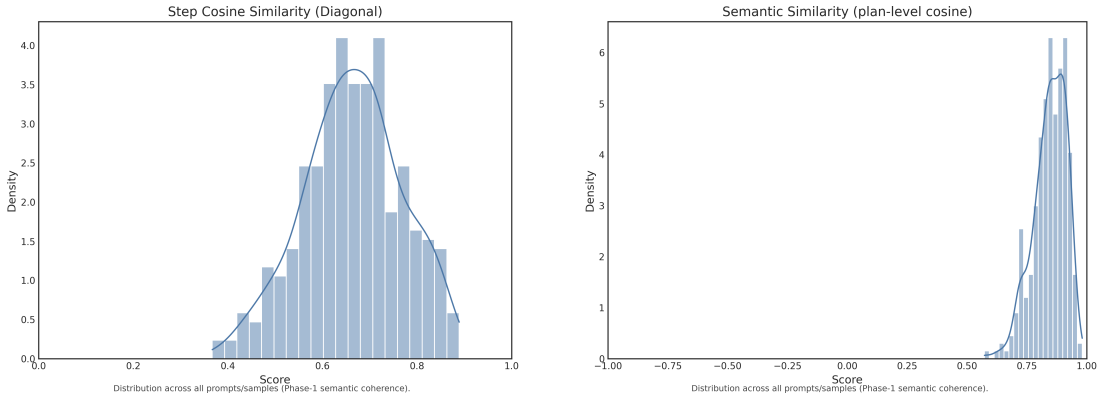


Figure 7.10: Phase A semantic distributions. Left: SSS concentrates near 0.65 and the tail of poorly aligned episodes contracts. Right: STS clusters around 0.85, evidencing strong plan-level agreement despite residual variance.

Behavioral observations. Qualitative inspection confirms the quantitative trends. Compared with the zero-shot regime, the fine-tuned model no longer emits refusal messages and rarely omits entire action categories, yet long-horizon tasks can still wander toward incorrect receptacles or skip optional clean-up steps. Figure 7.11 illustrates an ideal alignment, whereas Fig. 7.12 collects two counter-examples. These residual errors motivate the structured representation introduced in Phase B, which offers tighter control over ordering and grounding.

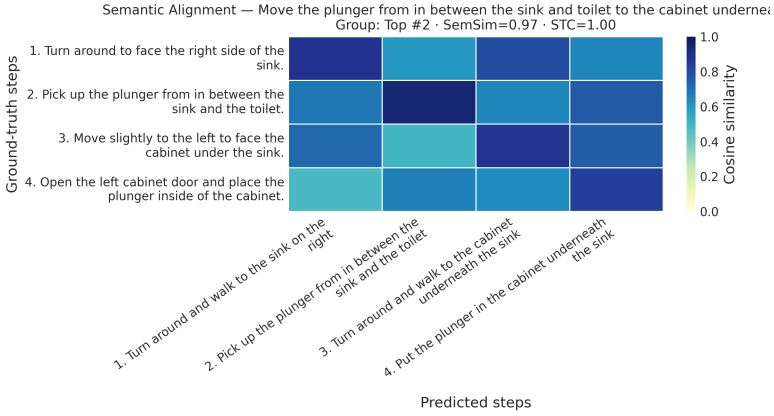
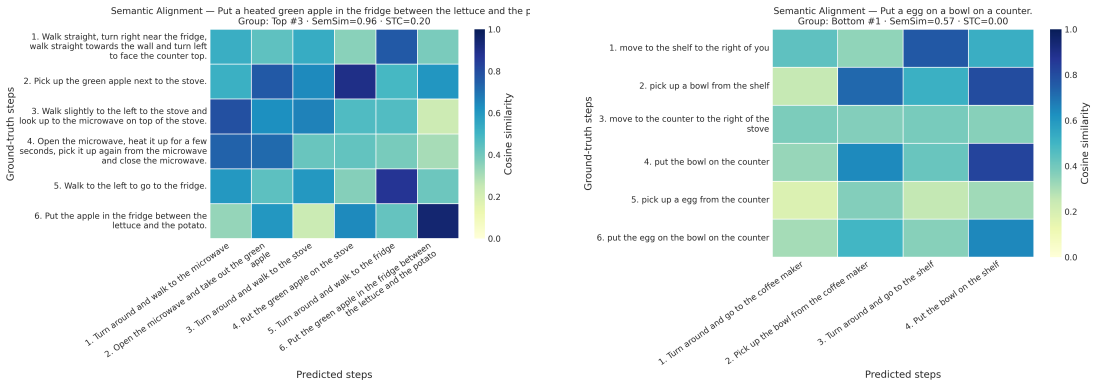


Figure 7.11: Phase A best-case heatmap: the plunger relocation task attains STS 0.97 and STC 1.00, with a clean diagonal and negligible off-diagonal bleed.



STS 0.96, STC 0.20 STS 0.57, STC 0.00

Figure 7.12: Representative failure modes. Left: high plan-level overlap but scrambled ordering. Right: the lowest-ranked sample, fluent yet instruction-mismatched.

To substantiate these findings, Table 7.7 reports three indicative episodes analyzed qualitatively. The first mirrors the reference plan almost exactly, whereas the other two expose residual ordering and grounding issues.

Table 7.7: Representative Phase A samples (fine-tuned model).

Instruction	Ground Truth Plan	Fine-tuned Model Output
Place the bat on the bed	Turn toward the doorway, collect the bat from the floor, walk back to the bed, place the bat on the blanket.	Turn around, walk to the wall, pick up the bat, walk to the bed, place the bat on the bed.
Move the corner bat to the bed	Navigate to the corner, grab the bat, return to the bed, drop the bat on the mattress.	Walk to the desk, pick up the bat, return it to the corner, leave it there.
Move a cup from the shelf to the desk	Approach the desk, take the cup from the upper shelf, move to the desk edge, set the cup down.	Walk to the bookshelf, pick up the cup, walk to the desk, put the cup on the shelf (reverse direction).

These samples reinforce the narrative drawn from the heatmaps: supervision delivers consistent, well-structured language plans, yet precise grounding and directionality still demand the action decomposition introduced in Phase B, where plans are emitted as concise sequences of discrete decisions rather than free-form prose.

7.6 Zero-Shot Prompt Comparison (Phase B)

Phase A showed that natural-language fine-tuning improves fluency, yet grounding mistakes remain. Phase B responds by adding a discrete action head aligned with the JSON representation defined in Chapter 4. The model now emits two synchronized outputs: a sentence-level plan that humans can read and a sequence of symbolic actions that a controller can execute.

The structured branch pursues three goals:

- preserve the readable planning interface introduced in Phase A;
- expose an executable recipe that downstream controllers can follow without extra parsing;
- diagnose grounding errors by aligning language tokens with explicit action arguments.

This dual output keeps the planning interface familiar while giving downstream modules a grounded bridge between text and control.

7.6.1 Quantitative and Qualitative Results

We compare five zero-shot prompting strategies adapted to the structured objective: One shot, Few shot, Visual focus, CoT, and RAR.[2] Each template preserves the natural-language question format introduced in Phase A but extends the output instructions so that the model must emit both the semicolon-delimited plan and a consistent discrete action list. Despite these additional constraints, all surviving prompts achieve perfect format compliance (1.0): outputs are well-formed JSON objects with the expected `nl_plan` and `discrete_action_list` fields, so downstream errors are driven by content rather than schema violations. We therefore report natural-language metrics, semantic alignment, discrete-action fidelity, and procedural diagnostics separately.

Lexical analysis. The natural-language results in Table 7.8 mirror the trends observed in Phase A (cf. Table 7.1): Visual focus and Few shot are the strongest prompting strategies on text-based metrics, with Visual focus slightly outperforming Few shot on both the average score and Token–F1. Notably, Phase B even achieves a modest improvement over the best Phase A setting: the average score for Visual focus reaches 0.350 here, compared with 0.342 for the best Phase A prompt (Few shot). By contrast, CoT and RAR tend to depress average scores because, in the dual-output setting, they may interleave fragments of the discrete action list or auxiliary boilerplate into the natural-language plan. Despite the more demanding objective in Phase B, Visual focus and Few shot remain effective at anchoring the model to a clean plan format; we defer to the semantic analysis for a discussion of persistent issues already highlighted in Chapter 7. As we will see in the qualitative analysis (Table 7.12), a side-by-side prompt comparison further illustrates these behaviors.

Table 7.8: Phase 2 natural-language metrics per prompt (mean ± standard deviation on TEST-UNSEEN). Best values in each column are in bold.

Prompt	Average score	BLEU	Token-F1	ROUGE-1	ROUGE-2	ROUGE-L
Visual focus	**0.316** ± 0.100	0.073 ± 0.084	**0.463** ± 0.108	**0.405** ± 0.133	**0.253** ± 0.104	**0.358** ± 0.117
Few shot	0.302 ± 0.083	0.074 ± 0.076	0.429 ± 0.085	0.399 ± 0.112	0.234 ± 0.088	0.352 ± 0.101
One shot	0.297 ± 0.081	**0.076** ± 0.076	0.416 ± 0.084	0.395 ± 0.110	0.228 ± 0.087	0.347 ± 0.097
CoT	0.046 ± 0.090	0.007 ± 0.034	0.099 ± 0.120	0.043 ± 0.120	0.025 ± 0.076	0.040 ± 0.109
RAR	0.026 ± 0.058	0.003 ± 0.026	0.071 ± 0.078	0.017 ± 0.074	0.009 ± 0.048	0.016 ± 0.069

[2]The "Style" template used in Phase A was removed because preliminary runs produced brittle JSON scaffolding and frequent format violations once discrete actions were requested.

Semantic analysis and heatmaps. Table 7.9 shows Few shot edging the other prompts on plan-level similarity (STS) and coverage, while Visual focus still claims the highest stepwise score (SSS = 0.535). Few shot and One shot remain clustered in the 0.45–0.47 band, confirming that improvements on step alignment are incremental. The distributions in Figure 7.14 confirm that Visual focus narrows variance but still delivers only modest absolute gains.

Compared with the Phase A baseline in Table 7.2, Visual focus incurs only a mild drop in plan-level similarity (STS 0.770 → 0.748) but further improves semantic alignment at the step level, raising the stepwise score from 0.494 to 0.535 and remaining the strongest prompt on SSS across both phases. In other words, it segments plans into discrete steps more reliably than the other prompts and aligns them more closely with the ground-truth boundaries, even if the absolute SSS value remains modest. This behavior is consistent with the Visual focus template itself: before emitting JSON, the prompt explicitly instructs the model to (i) look at all objects, their positions, and their states, (ii) notice changes between images that indicate progress, (iii) pay attention to spatial relationships and accessibility, and (iv) identify the exact locations and objects involved. These visual-grounding cues encourage the language branch to track per-step changes in the scene, improving stepwise semantic similarity despite the heavier decoding task.

Figure 7.15 illustrates these dynamics: the high-scoring episode (STS = 0.95, SSS = 0.66) follows the reference trajectory with only minor omissions, whereas the low-scoring case collapses into macro-action jargon (GotoLocation, PickupObject) inside the prose similarly to what happens in the CoT and RaR prompts. This prompt-driven mixing of the natural-language plan with the JSON format is what probably causes the steep decline in semantic scores when the model fails.

Table 7.9: Phase 2 semantic metrics per prompt (mean ± standard deviation).

Prompt	STS	SSS	Coverage@0.7	STC	AvgOffDiag
Few shot	$\mathbf{0.768} \pm 0.082$	0.474 ± 0.112	$\mathbf{0.512} \pm 0.219$	0.322 ± 0.369	0.446 ± 0.059
One shot	0.760 ± 0.084	0.453 ± 0.096	0.509 ± 0.212	0.262 ± 0.374	0.451 ± 0.056
Visual focus	0.748 ± 0.113	$\mathbf{0.535} \pm 0.127$	0.489 ± 0.240	$\mathbf{0.359} \pm 0.338$	$\mathbf{0.453} \pm 0.076$
CoT	0.219 ± 0.217	0.195 ± 0.131	0.056 ± 0.164	0.280 ± 0.355	0.156 ± 0.106
RAR	0.169 ± 0.145	0.165 ± 0.084	0.023 ± 0.109	0.269 ± 0.352	0.130 ± 0.069

Figure 7.13 makes the trade-off between language fluency and procedural alignment explicit. Visual focus sweeps the lexical axes while Few shot keeps semantic coverage competitive, underscoring that no single prompt dominates both radar charts.

The distribution plots in Figure 7.14 confirm a long tail of under-aligned plans: most samples cluster around moderate similarity, but a non-negligible mass drops near zero, signaling poor grounding.

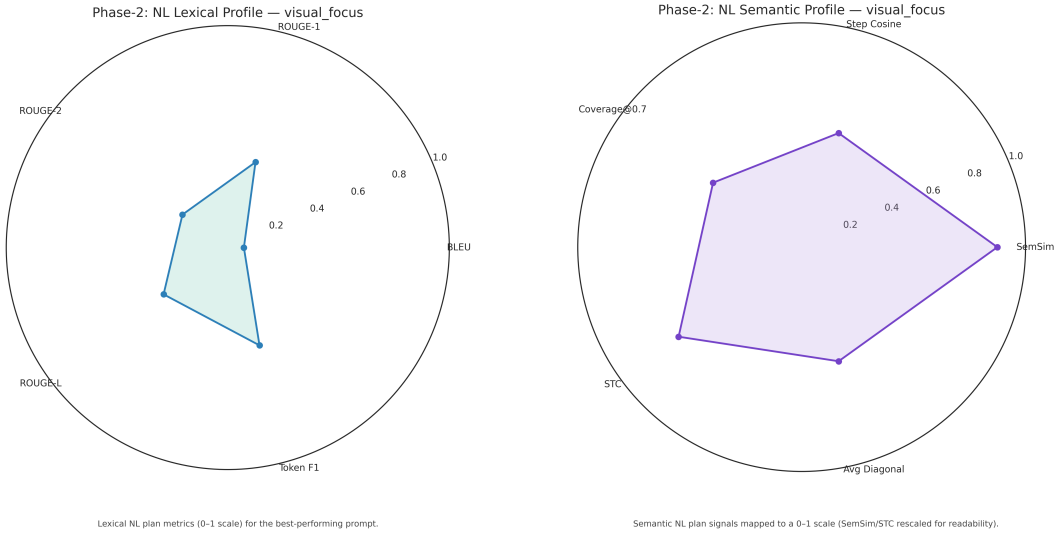


Figure 7.13: Composite radar plots. Left: natural-language metrics (Average score, BLEU, Token-F1, ROUGE). Right: semantic metrics (STS, Step Cosine Similarity, Coverage@0.7, STC).

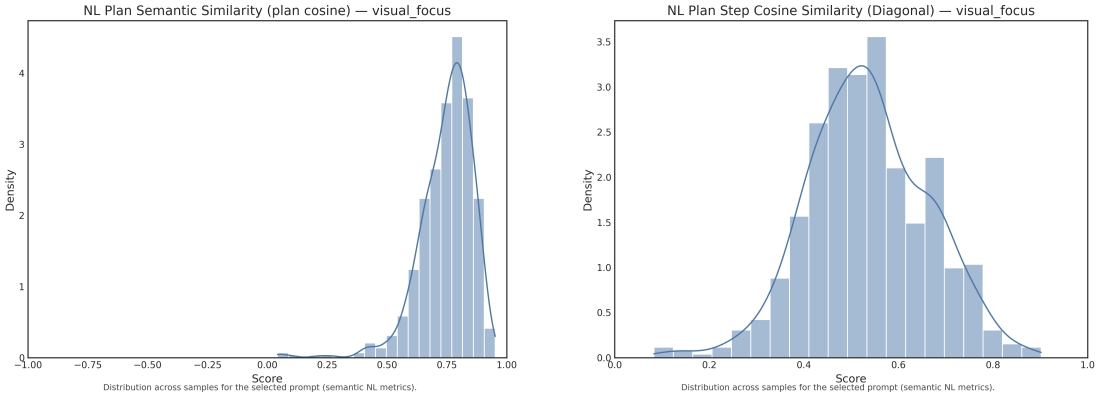


Figure 7.14: Distributions of NL plan semantic similarity (left) and step cosine similarity (right) for the Visual focus prompt.

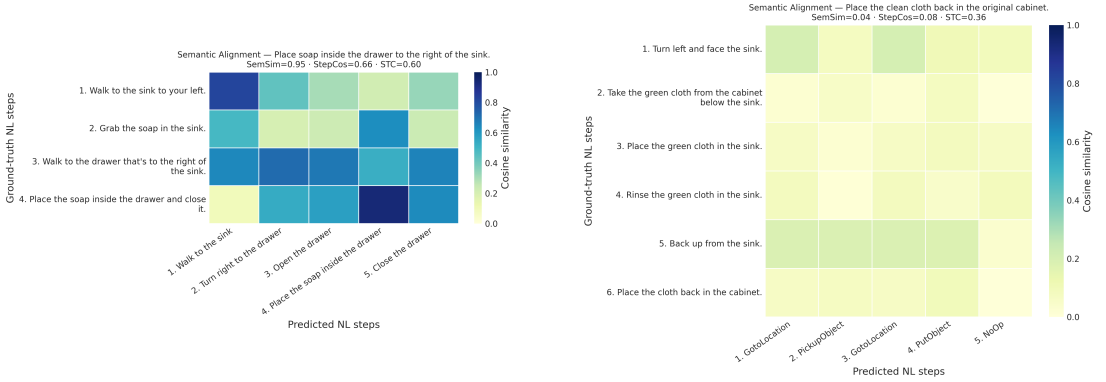


Figure 7.15: Phase B semantic heatmaps for the Visual focus prompt. Left: high alignment (STS = 0.95, SSS = 0.66) with a clear diagonal. Right: failure case (STS = 0.04, SSS = 0.08) where the text plan collapses into macro actions.

Discrete-action analysis. Table 7.10 highlights how prompt design steers the discrete head. CoT attains the highest action score (0.452), with strong action-type (0.753) and sequence similarity (0.431), despite delivering some of the weakest natural-language plans in Table 7.8. The chain-of-thought template explicitly scaffolds intermediate reasoning prior to emitting the JSON block, which appears to stabilise the symbolic branch and encourages the model to enumerate canonical transitions even when the accompanying prose degenerates. RAR follows closely on structural overlap (type 0.764, sequence 0.438) but its argument similarity collapses (0.123) because the prompt favours templated responses that omit contextual details. Crucially, neither CoT nor RAR provide in-context examples of how objects and locations are named in ALFRED; once the model has visually identified an entity, it therefore defaults to its own vocabulary rather than the ground-truth labels, further depressing argument similarity scores. More descriptive prompts such as One shot and Few shot provide richer grounding for arguments (up to 0.240) yet trail the structured templates on ordering fidelity. Visual focus sits between these extremes: visual hints help recover argument tokens, but the additional textual burden reduces type and sequence agreement compared with CoT.

This mirrors the Phase 2 prompt scripts: CoT encourages step-by-step reasoning before emitting JSON, which helps preserve action order, but none of the templates explicitly supervise argument vocabulary, so grounding quality remains fragile in the zero-shot regime. The lack of a dedicated argument-level fine-tuning step limits absolute performance overall, as across prompts the similarity rarely exceeds 0.25, underscoring that object and receptacle grounding remains the primary failure mode for the discrete branch.

Figure 7.16 reinforces this split: CoT and RAR deliver compact, well-aligned

Table 7.10: Phase 2 discrete-action metrics per prompt (mean ± standard deviation).

Prompt	Type sim.	Sequence sim.	Argument sim.	Action score
One shot	0.493 ± 0.142	0.310 ± 0.205	$\mathbf{0.240} \pm 0.192$	0.348 ± 0.117
Few shot	0.516 ± 0.172	0.341 ± 0.227	0.230 ± 0.165	0.362 ± 0.126
RAR	$\mathbf{0.764} \pm 0.166$	$\mathbf{0.438} \pm 0.159$	0.123 ± 0.159	0.442 ± 0.099
CoT	0.753 ± 0.171	0.431 ± 0.166	0.171 ± 0.166	$\mathbf{0.452} \pm 0.105$
Visual focus	0.581 ± 0.185	0.403 ± 0.233	0.194 ± 0.160	0.393 ± 0.138

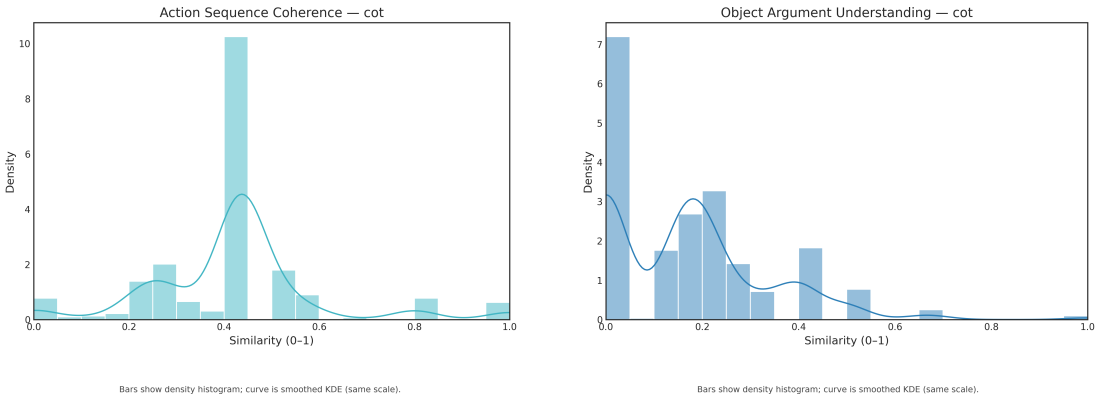


Figure 7.16: Action sequence coherence (left) and object argument understanding (right) across prompts. Both plots show Few shot and Visual focus balancing structural fidelity with partial gains on object grounding, whereas CoT and RAR favour strict order over argument coverage.

sequences yet fail to recover argument vocabulary, while Visual focus and Few shot trade some ordering precision for richer object grounding.

Joint perspective. Figure 7.17 summarises the trade-offs across prompts. Structured templates such as CoT and RAR secure high action scores (0.45 and 0.44) but collapse on natural-language quality (0.05 and 0.03), reflecting the lexical/semantic issues already discussed. Conversely, Visual focus leads the language branch (NL comprehensive ≈ 0.32) while maintaining solid discrete actions (0.39), and Few shot offers the most balanced compromise (0.30 vs 0.36). This dispersion confirms that a single aggregated score would hide important prompt-specific behaviour; evaluating lexical, semantic, and action metrics separately is crucial when dual outputs are required. Inspecting the Phase 2 prompt templates shows why: CoT and RAR emphasise internal reasoning steps before emitting JSON, prioritising structural fidelity, whereas Visual focus and One/Few-shot devote more budget to descriptive cues, favouring the natural-language plan.

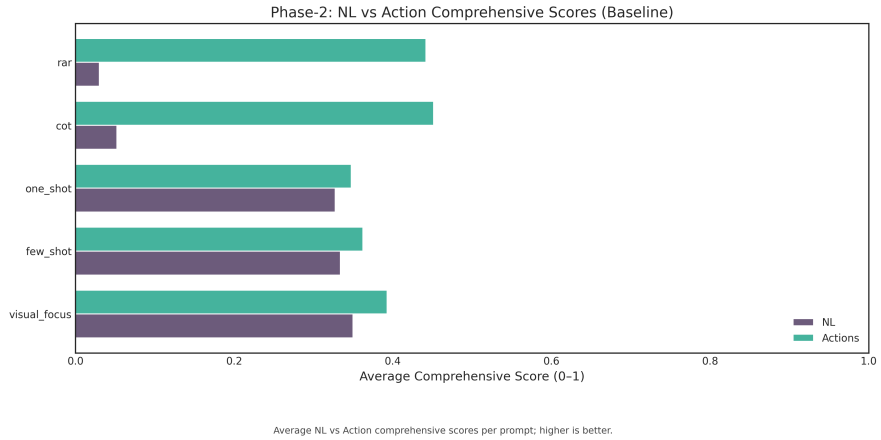


Figure 7.17: Ranking prompts by the dual comprehensive score (average of natural-language and action branches).

Efficiency and verbosity. Inference slows down in Phase B due to the added JSON output, with per-episode times rising to about 32–45 s. Reasoning-style prompts (RAR, CoT) become the fastest and most compact, though their brevity largely reflects plans that collapse into macro primitives rather than full semicolon-separated steps. One-Shot and Few-Shot show the highest latency and longest outputs because they retain a detailed step-by-step structure, while Visual Focus falls in between, combining moderate length with more grounded descriptions.

Procedural Diagnostics We use the Argument Consistency Index (ACI), a single measure that summarises how consistently the model recovers object, container, location, and type arguments, to inspect procedural fidelity. Table 7.11 contrasts CoT and Few-shot: CoT keeps traces short and structurally aligned (high canonical-pair coverage, low extra steps) but misses actions entirely, whereas Few-shot covers more containers yet over-generates and still omits required moves. Figure 7.18 complements this view: the KDE of ACI values is bimodal, with peaks near 0.1 and 0.6, signalling unstable grounding. Scenario-wise, "Pick Cool Then Place in Receptacle" sits just below 0.6 because its two-stage routine is predictable, while "Pick and Place with Movable Receptacle" falls toward 0.3 as both object and receptacle shift. Even the better-performing routines therefore underline that receptacle reasoning remains the main bottleneck.

Table 7.11: Key procedural metrics for CoT and Few-shot prompts.

Metric	CoT	Visual Focus
Argument Consistency Index (ACI)	0.35	0.34
Object Coverage F1	0.57	0.53
Container Coverage F1	0.18	0.20
Location Coverage F1	0.18	0.18
Type-Bucket Arg Micro-F1	0.30	0.28
Canonical Pair Coverage	0.80	0.67
Movement Precision	0.22	0.21
Movement Recall	0.14	0.14
Extra Steps (Pred − GT ≥ 0)	0.14	0.20
Missing Steps (GT − Pred ≥ 0)	2.91	3.01

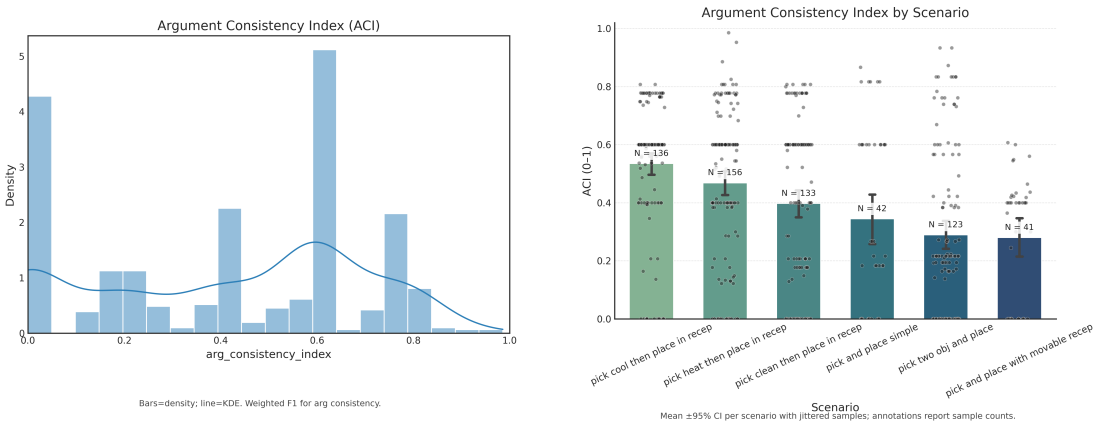


Figure 7.18: Argument Consistency Index diagnostics. Left: overall distribution across prompts. Right: scenario-level breakdown showing that container-heavy tasks amplify variance.

Qualitative synthesis. Table 7.12 examines a representative episode ("move two bars of soap to the bin") and reflects the broader zero-shot trends. CoT and RAR,despite competitive discrete scores, still output macro-level placeholders in the natural-language plan and fail to adopt the expected semicolon-separated structure, as we already analyzed in the previous sections; their JSON traces are similarly compact and often omit the second transfer. By contrast, the descriptive prompts (One-Shot, Few-Shot, Visual Focus) recover both moves in plain English. The first two remain concise and procedural, whereas Visual Focus introduces color and object descriptors that enhance grounding at the cost of longer narratives. Overall, zero-shot prompting helps stabilize formatting and preserves structural coherence, but the argument layer remains fragile; targeted adaptation—via fine-tuning, constrained decoding, or richer supervision—is required to mitigate missing steps and improve argument fidelity.

Instruction	Move two bars of soap from the sink to a small black bin.
Ground Truth	**NL plan**: Turn left and walk towards the wall, then hang a right and walk up to the sink; pick up the rightmost bar of soap; walk to the bin; drop it; return to the sink; pick up the second soap; walk back to the bin; place it inside. **Discrete actions**: GotoLocation(sink basin) → PickupObject(soapbar) → GotoLocation(garbagecan) → PutObject(soapbar, garbagecan) → GotoLocation(sink basin) → PickupObject(soapbar) → GotoLocation(garbagecan) → PutObject(soapbar, garbagecan) → NoOp().
One-Shot	**NL plan**: Walk to the sink; open the sink; take a bar of soap from the sink; place the bar of soap in the bin; take another bar of soap from the sink; place the bar of soap in the bin; close the sink. **Discrete actions**: GotoLocation(sink) → OpenObject(sink) → PickupObject(soap) → PutObject(soap, bin) → PickupObject(soap) → PutObject(soap, bin) → CloseObject(sink).
Few-Shot	**NL plan**: Walk to the sink; grab the soap on the sink; walk to the bin; put the soap in the bin; grab the other soap on the sink; walk to the bin; put the soap in the bin. **Discrete actions**: GotoLocation(sink) → PickupObject(soap) → GotoLocation(bin) → PutObject(soap, bin) → GotoLocation(sink) → PickupObject(soap) → GotoLocation(bin) → PutObject(soap, bin) → NoOp().
CoT	**NL plan**: GotoLocation; PickupObject; GotoLocation; PutObject; NoOp. **Discrete actions**: GotoLocation(Sink) → PickupObject(Soap) → GotoLocation(Black Bin) → PutObject(Soap, Black Bin) → NoOp().
RAR	**NL plan**: GotoLocation; PickupObject; GotoLocation; PutObject; NoOp. **Discrete actions**: GotoLocation(Sink) → PickupObject(Soap) → GotoLocation(Black Bin) → PutObject(Soap, Black Bin) → NoOp().
Visual Focus	**NL plan**: Walk to the sink; pick up the pink bar of soap on the sink; walk to the black bin; put the pink bar of soap on the black bin; walk to the sink; pick up the yellow bar of soap on the sink; walk to the black bin; put the yellow bar of soap on the black bin. **Discrete actions**: GotoLocation(sink) → PickupObject(pink bar of soap) → GotoLocation(black bin) → PutObject(pink bar of soap, black bin) → GotoLocation(sink) → PickupObject(yellow bar of soap) → GotoLocation(black bin) → PutObject(yellow bar of soap, black bin).

Table 7.12: Qualitative comparison across prompts for the "move two bars of soap to the bin" task. Natural-language plans and discrete action traces highlight how each prompt balances fluency and grounded execution on the same episode.

7.7 Fine-Tuned Model on ALFRED – Structured Planning with Discrete Actions (Phase B)

Fine-tuning in Phase B targets the failure modes highlighted by the zero-shot prompts: argument slots remained brittle, discrete traces often skipped actions, and semicolon-delimited plans occasionally devolved into macro tokens. The contextual fine-tuning recipe introduced in Chapter 6 is therefore applied to the Phase B objective, with the vision encoder and base LLM kept frozen while only the multimodal projector and LoRA heads are updated. Evaluation reuses the metric suite from Section 7.2, now interpreted for the dual-output setting (natural-language branch plus discrete-action branch).

7.7.1 Training Configuration and Loss Trends

The contextual run started from the configuration per-device batch size 1, gradient accumulation 8, a cosine schedule with base rate 1×10^{-6} (warm-up ratio 0.03), Deepspeed ZeRO-2, FP16, and gradient checkpointing, while both the vision tower and LLM weights remained frozen. Early logs revealed that such a large accumulation factor smoothed gradients excessively; therefore, around step 2.5k we switched to batch size 2 with accumulation 2 (keeping the effective batch at 4). Figure 7.19 shows the corresponding loss trace: immediately after the change the curve drops sharply from roughly 3 to 1.5 and resumes a steady descent, confirming that the smaller accumulation unlocked faster optimization without destabilizing training.

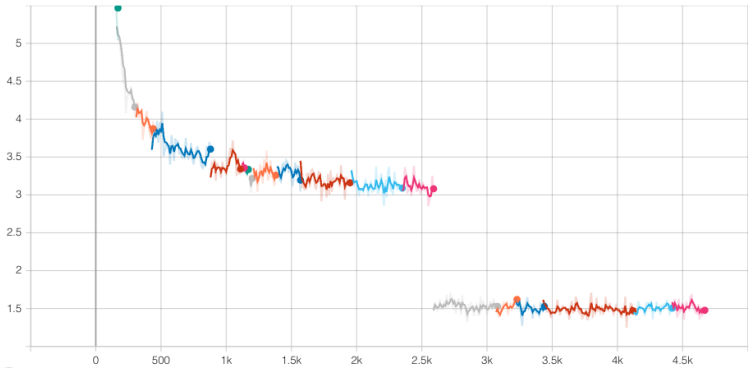


Figure 7.19: Fine-tuning loss for the contextual planner. The discontinuity near step 2.5k corresponds to reducing gradient accumulation from 4 to 2, which yields a noticeably steeper descent.

After the adjustment, the loss stabilizes around 1.4–1.6 with mild oscillations caused by checkpoint restarts, matching the stable optimization observed in Phase A

(Figure 7.8).

In principle, the effective batch size should be held constant across a run to keep the optimization regime comparable; here hardware limits forced us to reduce it mid-training. Despite this constraint, the model not only exhibited the sharp loss break documented above, but also proved to be the best-performing variant across our experiments, after several prior runs that had to be discarded. This suggests that, for our setting, smaller accumulation (and hence fresher gradients) was preferable to a larger, smoother batch.

7.7.2 Prompting Strategy and Grounding Techniques

We adopt a compact chat template to improve visual grounding, task focus, and output-format consistency during fine-tuning. The design combines three complementary ingredients.

Temporal grounding via numbered observations. Each input frame is introduced by a textual marker (Observation 1, Observation 2, ...), followed by the corresponding image. These markers act as anchors for the cross-attention layers: each "Observation i:" is a text prefix immediately preceding the visual tokens of the next image, so visual blocks are separated by distinct linguistic contexts, and subsequent text can attend to the relevant block based on the index. Explicit frame indexing has shown similar benefits in Video-LLaMA [27].

Semantic grounding via object-list dropout. With fixed probability $p = 0.15$, we append a natural-language list of objects available in the scene drawn from ALFRED annotations (e.g., "Available objects in the scene: mug, plate, table"). When present, this list anchors visual entities to text; when omitted, the model must rely solely on images. This controlled stochasticity acts as input dropout, preventing over-reliance on textual hints and improving cross-modal robustness.

Task anchoring through a fixed system prompt. A concise system instruction specifies the role and enforces the output schema: two fields (nl_plan and discrete action list) returned only in JSON, with actions chosen from the ALFRED inventory (GotoLocation, PickupObject, PutObject, ...). System-level constraints of this kind stabilise instruction-tuned VLMs [28, 29], reducing off-task generation and format drift.

The prompting template combines three key elements: observation numbering, object-list dropout, and a fixed system prompt. Observation numbering provides a lightweight temporal reference across images; the probabilistic object list regularizes visual–textual conditioning; and the system prompt enforces a consistent output format. Together, these design choices improved JSON compliance, grounding

accuracy, and argument recovery without relying on the object list being always present. The full chat template specification and implementation are available in the project's GitHub repository.

7.7.3 Quantitative and Qualitative Results

We evaluate the fine-tuned structured planner on the ALFRED test-unseen split using the same metrics defined in Section 7.2 and the experimental setup of Chapter 6. The analysis is organized along five axes: natural-language metrics, plan-level and stepwise semantics, discrete-action similarity (types, order, arguments), procedural diagnostics (ACI and coverage), and qualitative examples.

Efficiency and decoding stability. Token counts remain moderate on average (106.6 ± 35.9 tokens), yet Figure 7.20 exposes a heavy-tailed compute profile. Most episodes finish within a minute, but a non-trivial subset stretches beyond 10 minutes, yielding an inference-time mean of 129.0 s with a large standard deviation (± 189.5 s). The shallow correlation ($\rho \approx 0.06$) between length and latency suggests that the spikes stem from decoder instabilities rather than sheer output size, often when the model struggles to terminate JSON generation. This variance is a practical limitation: downstream systems need guardrails (time-outs, constrained decoding, partial responses) to avoid stalling on the worst cases, and future work should target steadier decoding without sacrificing plan quality. From a memory perspective, instantiating the base LLaMA 3.2 Vision–Instruct checkpoint in our ALFRED Phase B setup already occupies roughly 23 183 MiB on the 80 GiB GPU, and the fine-tuned planner peaks at about 35 641 MiB (43.5% of device memory) during inference with 15-frame inputs. In Chapter 9 we therefore investigate whether 4-bit bnb-int4 quantization can deliver a more efficient planner—reducing memory footprint and latency while preserving acceptable plan quality.

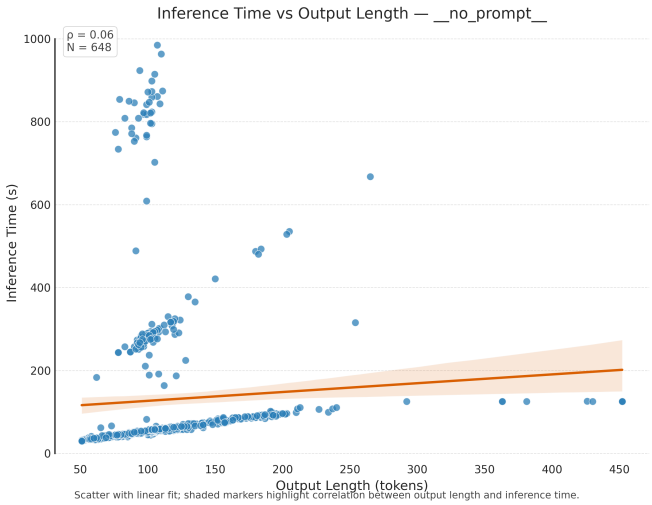


Figure 7.20: Phase B fine-tuned planner: inference time vs. output length (tokens). The weak correlation ($\rho \approx 0.06$) and heavy tail of slow episodes highlight that latency spikes stem from decoding instabilities rather than sheer plan length, motivating the use of guardrails and more efficient decoding schemes.

Lexical analysis. The fine-tuned Phase B model demonstrates substantial improvements across all lexical evaluation metrics on the test-unseen split, as summarised in Table 7.13. The average composite score reaches 0.522, representing a notable advancement of +0.172 points over the best-performing zero-shot configuration (Visual focus at 0.350) and a +0.068 gain relative to the Phase A fine-tuned baseline (0.454). This performance elevation reflects the model's enhanced capacity to generate coherent, well-structured natural language plans while maintaining strict adherence to the required JSON output format. The improvements are particularly pronounced in surface-level similarity metrics, with BLEU scores rising to 0.290, a substantial increase of +0.150 points compared to Phase A fine-tuning and a remarkable +0.217 improvement over Phase B zero-shot performance. Similarly, Token-F1 achieves 0.629, indicating strong lexical overlap with reference plans, while the ROUGE family of metrics demonstrates consistent gains: ROUGE-1 reaches 0.618, ROUGE-2 attains 0.379, and ROUGE-L consolidates at 0.532. The latter metric is particularly significant, as it captures global structural alignment and shows improvements of +0.174 relative to zero-shot Phase B and +0.069 compared to Phase A fine-tuning. The rise of the average score above 0.5 indicates that the model has learned to appropriately sequence contextual information including task instructions, visual cues, and object references. However, perfect natural language planning remains challenging for complex multi-step tasks requiring reasoning over 15 egocentric frames, particularly under stringent metrics like BLEU. The model's perfect format compliance (1.0) alongside these lexical gains demonstrates the effectiveness of structured fine-tuning.

Table 7.13: Phase B lexical metrics (fine-tuned model, TEST-UNSEEN, mean ± standard deviation).

Metric	Mean ± Std.
Average Score	0.522 ± 0.083
BLEU	0.290 ± 0.110
Token F1	0.629 ± 0.083
ROUGE-1	0.618 ± 0.087
ROUGE-2	0.379 ± 0.095
ROUGE-L	0.532 ± 0.088

Semantic analysis and heatmaps. Relative to Phase A (Table 7.6), the fine-tuned model exhibits consistent, albeit moderate, gains on all semantic indicators (Table 7.14). The largest improvement concerns temporal coherence: STC rises from 0.474 to 0.664, signaling a stronger preservation of step order between predicted and reference plans. Plan-level similarity also slightly improves (STS 0.861 vs.

0.845), as well as stepwise alignment (SSS 0.742 vs. 0.664). The distributional view in Figure 7.21 corroborates this trend: compared with Phase A (Figure 7.10, right panel), the SSS curve is clearly displaced to the right, indicating a higher mass of well-aligned steps while maintaining a comparable overall shape.

Heatmap evidence reflects the same pattern. In Figure 7.22, the left panel shows an almost perfect diagonal (SSS close to 1, STC near 1) for the instruction "Place a rinsed egg in the microwave", where every ground-truth step finds a precise counterpart. The right panel, drawn from one of the lowest-scoring episodes, illustrates residual failure modes: occasional hallucinations or off-topic turns (e.g., turning to face furniture unrelated to the goal) introduce off-diagonal mass. Nevertheless, the plan is rarely degenerate; most steps still lie near the diagonal and are not omitted.

Taken together, these results indicate that fine-tuning improves semantic faithfulness and temporal stability of the natural-language plans, though not uniformly across scenarios. Remaining errors appear linked primarily to visual ambiguity in the 15-frame context and to mild stylistic biases (for instance, stereotyped openings such as "Turn around and ..."). While such biases can be mitigated with targeted supervision or decoding constraints, the present level of semantic reliability is sufficient for our purposes, especially given the strong discrete-action quality reported below.

Table 7.14: Phase B semantic metrics (fine-tuned model, mean ± standard deviation on TEST-UNSEEN).

Metric	Mean ± Std.
STS	0.861 ± 0.068
SSS	0.742 ± 0.093
Coverage@0.7	0.806 ± 0.170
STC	0.664 ± 0.298
AvgOffDiag	0.517 ± 0.058

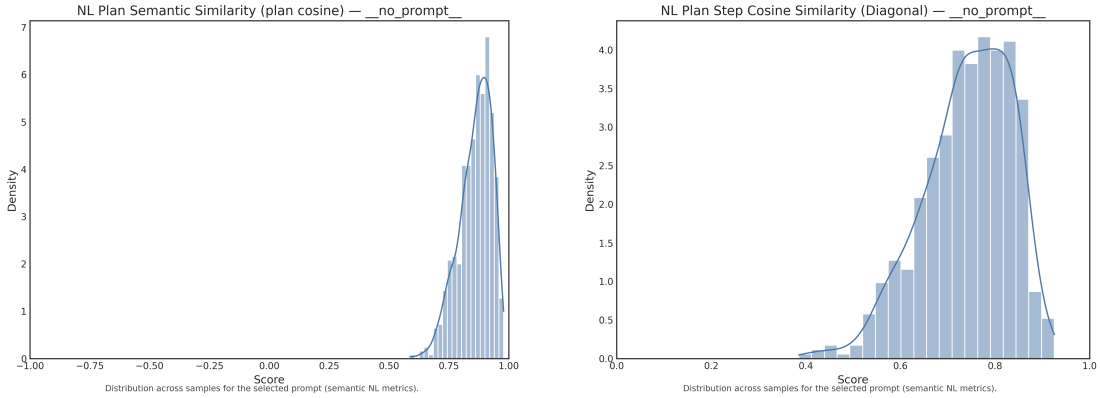


Figure 7.21: Fine-tuned Phase B distributions: STS (left) and SSS (right). Plan-level similarity is tightly concentrated near 0.85, while the stepwise curve shifts markedly to the right compared with zero-shot baselines, confirming that fine-tuning narrows variance and increases the proportion of well-aligned steps.

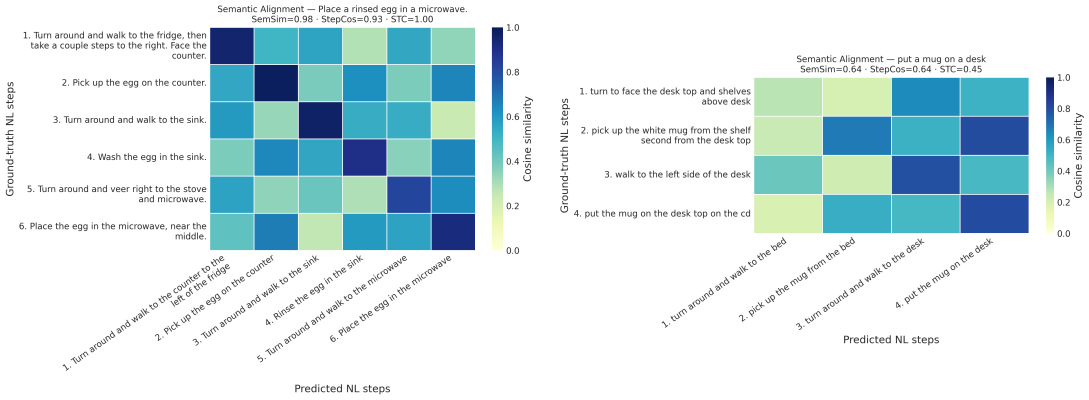


Figure 7.22: Fine-tuned Phase B semantic heatmaps: best case (left) and difficult case (right). The near-perfect diagonal on the left illustrates how most ground-truth steps find clean matches after fine-tuning, whereas the right-hand failure case still shows off-diagonal mass triggered by hallucinated turns and receptacle errors, highlighting the residual scenarios where grounding breaks down.

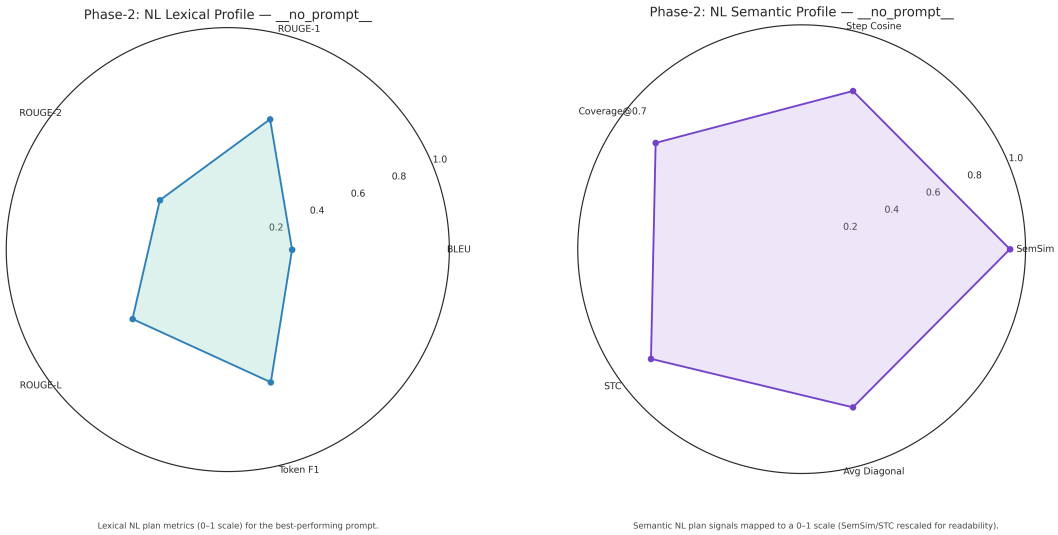


Figure 7.23: Fine-tuned Phase B radars: lexical (left) and semantic (right).

Discrete-action analysis and procedural diagnostics. The discrete-action branch exhibits the most remarkable progress across all experimental phases. As shown in Figure 7.25, the fine-tuned radar forms an almost perfect square, indicating that type, sequence, argument, and overall action accuracy have converged to comparable, near-optimal levels around 0.9. This geometric symmetry contrasts sharply with the irregular and unbalanced profiles observed in the zero-shot prompts, where sequence fidelity and argument grounding often diverged. Fine-tuning thus yields a highly stable and coherent controller that reproduces the symbolic structure and its corresponding arguments with exceptional consistency.

Quantitatively, Table 7.15 confirms this convergence: type similarity reaches 0.955, sequence similarity 0.883, argument similarity 0.865, and the combined action score 0.901. These values mark a decisive leap from the zero-shot regime, where scores rarely exceeded 0.45. The improvement is directly linked to the contextual fine-tuning recipe described in Chapter 6, in which the Prompting Strategy and Grounding Techniques (Section 7.7.1), notably observation numbering, object-list dropout, and a fixed system instruction, jointly enhance the model's temporal grounding, argument recovery, and JSON stability. By coupling architectural alignment with controlled prompting, the model learns to anchor every symbolic step to visual evidence while preserving strict syntactic compliance.

Procedural diagnostics reinforce this interpretation. The Argument Consistency Index (ACI) averages 0.901 (Table 7.16), while canonical-pair coverage rises to 0.969, indicating that canonical transitions such as GotoLocation → PickupObject and OpenObject → PutObject are faithfully recovered. Object, container, and

location coverage scores remain consistently above 0.88, and the frequency-aware Type-Bucket Micro-F1 (0.877) confirms that the model also preserves repetition patterns, not merely unique arguments. The ACI distribution in Figure 7.26 collapses into a sharp unimodal peak near 0.9, in contrast with the wide bimodal spread seen in the zero-shot regime, demonstrating both structural and procedural regularity. Scenario-wise breakdowns further show that even container-intensive routines, previously unstable, now achieve uniform consistency.

In sum, the combination of contextual fine-tuning and structured prompting transforms the discrete-action branch from a brittle symbolic emitter into a reliable procedural planner. The nearly symmetric radar profile and concentrated ACI peak testify to a model that has internalized the causal and temporal dependencies underlying ALFRED-style plans. Although there remains scope for refinement, particularly in fine-grained argument precision and over-generation control, the results already demonstrate a decisive leap in procedural reliability after a single epoch of adaptation. The following section on Qualitative synthesis provides concrete visual and textual evidence of this behaviour, showcasing how the model now generates grounded, executable discrete plans with unprecedented coherence.

We report discrete-action similarities in Table 7.15 and summarise structure and grounding in Figure 7.24.

Table 7.15: Phase B discrete-action metrics (fine-tuned model, TEST-UNSEEN, mean ± standard deviation).

Metric	Mean ± Std.
Type sim.	0.955 ± 0.093
Sequence sim.	0.883 ± 0.219
Argument sim.	0.865 ± 0.192
Action score	0.901 ± 0.121

Table 7.16: Phase B procedural diagnostics (fine-tuned model, TEST-UNSEEN; mean ± std).

ACI	0.901 ± 0.146
Canonical Pair Coverage	0.969 ± 0.106
Object Coverage F1	0.927 ± 0.226
Container Coverage F1	0.890 ± 0.177
Location Coverage F1	0.884 ± 0.186
Type Micro-F1 (Obj/Cont/Loc)	0.877 ± 0.159
Movement Precision	0.858 ± 0.196
Movement Recall	0.865 ± 0.200
Extra Steps	0.310 ± 0.837
Missing Steps	0.260 ± 0.890

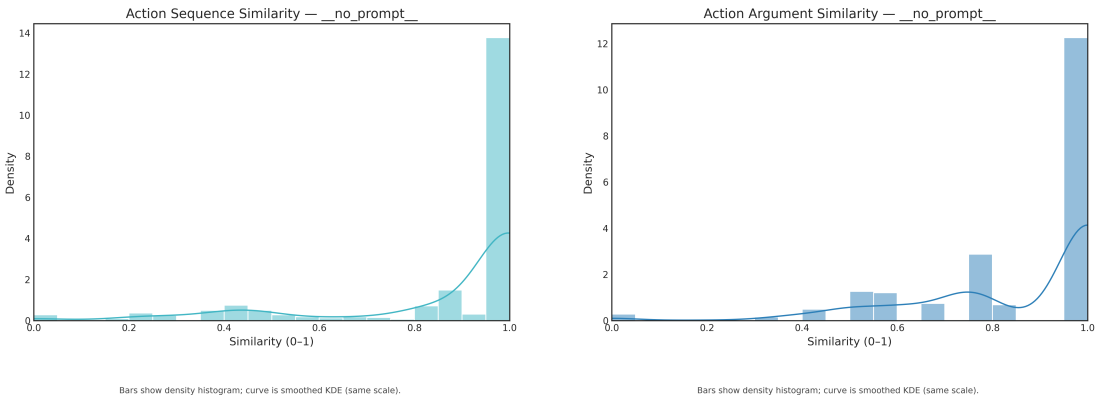


Figure 7.24: Fine-tuned Phase B: sequence similarity (left) and argument similarity (right) distributions. Both histograms collapse near 1.0 after fine-tuning, showing that the discrete branch now recovers the correct action order and object arguments for almost every episode, with only a few low-similarity outliers remaining.

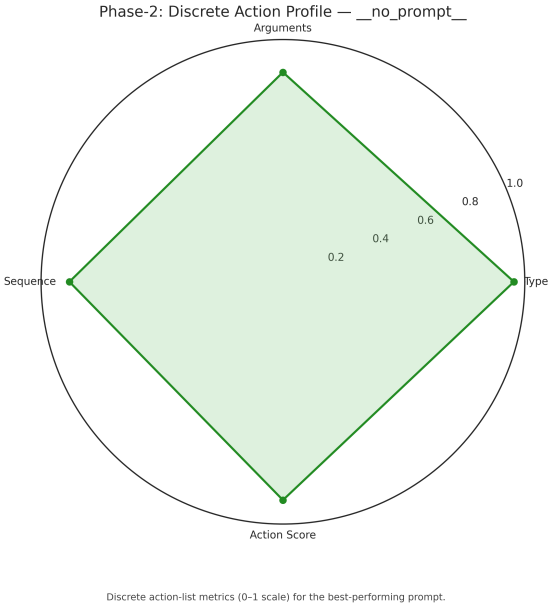


Figure 7.25: Fine-tuned Phase B discrete-action radar.

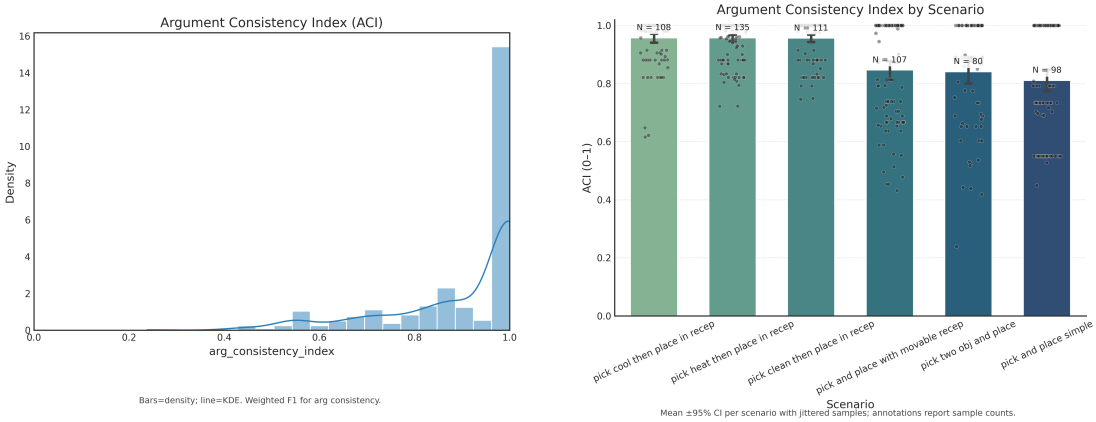


Figure 7.26: Fine-tuned Phase B: overall ACI distribution (left) and ACI by scenario (right).

Qualitative synthesis. Table 7.17 reports representative Phase B samples pairing the natural-language plan with the corresponding discrete action list.

A recurring pattern observable across several fine-tuned samples is the systematic presence of the redundant prefix "turn around and walk. . . " at the beginning of a several natural-language plan. This stylistic artifact reflects a dataset-induced bias rather than a genuine procedural need. Since many ALFRED demonstrations start from an egocentric viewpoint where the initial step often involves a re-orientation action (e.g., "turn left," "turn right," "turn around"), the fine-tuned model has internalized these orientation verbs as the most probable opening transition. Consequently, even when the initial rotation is unnecessary in context, the model tends to include it by default, revealing a strong prior over directional motion rather than scene-specific reasoning.

Despite this redundant phrasing, the underlying procedural structure remains coherent and executable. The model consistently preserves the causal and spatial dependencies required to complete the task, as evidenced by its near-perfect discrete-action precision. In both examples, the symbolic traces align closely with the reference plans: every action type, sequence order, and argument pairing is correctly recovered, demonstrating that fine-tuning has yielded a controller capable of executing well-grounded and temporally consistent procedures.

The overall comprehensive score computed as the mean between the natural-language and discrete-action branches reaches approximately **0.81**, combining a high discrete-action accuracy (Action Score ≈ 0.90) with solid lexical alignment (≈ 0.72). This confirms that, while the model occasionally over-generalises frequent motion patterns inherited from training data, it nevertheless maintains excellent procedural fidelity. In particular the discrete branch exhibits high precision and stability, reflecting a robust internalization of task structure and a clear ability to translate visual context and language instructions into precise, executable symbolic actions.

Instruction	Prediction
put a bowl and a pencil between the books on the desk	**NL plan (GT)**: go to the desk; pick up the pencil from the desk; re-approach the desk where the bowl is; put the pencil in the bowl; pick up the bowl with the pencil; take the bowl to the right side of the desk; set the bowl on the desk between the two rows of books on the bottom shelf. **Discrete actions (GT)**: GotoLocation(desk) → PickupObject(pencil) → GotoLocation(bowl) → PutObject(pencil, bowl) → PickupObject(bowl) → GotoLocation(shelf) → PutObject(bowl, shelf) → NoOp(). **NL plan (Pred)**: turn around and walk to the desk; pick up the pencil on the desk; turn around and walk to the wall; put the pencil in the bowl on the desk; pick up the bowl on the desk; turn around and walk to the desk; put the bowl between the books on the desk. **Discrete actions (Pred)**: GotoLocation(desk) → PickupObject(pencil) → GotoLocation(bowl) → PutObject(pencil, bowl) → PickupObject(bowl) → GotoLocation(desk) → PutObject(bowl, desk) → NoOp().
put green cup and knife into sink basin	**NL plan (GT)**: walk to face counter behind sink; pick up butter knife from under sponge on counter; walk to face counter to left of stove; put knife into green cup on the left on the counter; pick up the green cup from counter; turn to face the kitchen sink; put green cup into sink basin. **Discrete actions (GT)**: GotoLocation(countertop) → PickupObject(butterknife) → GotoLocation(cup) → PutObject(butterknife, cup) → PickupObject(cup) → GotoLocation(sinkbasin) → PutObject(cup, sinkbasin) → NoOp(). **NL plan (Pred)**: turn around and walk to the counter to the right of the sink; pick up the knife from the counter; turn around and walk to the counter to the right of the sink; put the knife into the green cup on the counter; pick up the green cup from the counter; turn around and walk to the sink; put the green cup into the sink basin. **Discrete actions (Pred)**: GotoLocation(countertop) → PickupObject(butterknife) → GotoLocation(cup) → PutObject(butterknife, cup) → PickupObject(cup) → GotoLocation(sinkbasin) → PutObject(cup, sinkbasin) → NoOp().

Table 7.17: Representative Phase B samples (fine-tuned model). Each row shows the instruction, the associated natural-language plan and the discrete action sequence (GT for the first row; model prediction for the second).

Chapter 8

Open X-Embodiment: Evaluation and Results

8.1 Introduction

In this chapter, we investigate the extension of the LLaMA 3.2 Vision 11B model toward low-level motor control using the Open X-Embodiment dataset. Unlike the previous ALFRED experiments, where the output consisted of structured symbolic plans or natural-language instructions, the model here is required to generate continuous control actions represented as discretized 8-token vectors. Each action vector encodes the physical command issued to the robot at a given timestep, enabling a direct link between visual perception and actuation.

All experiments in this phase are conducted in an offline setup, meaning that the model is not deployed in a closed-loop robotic environment but rather evaluated on pre-recorded trajectories. This choice is primarily motivated by computational constraints: training a vision-language model on the complete RT-1 subset of the Open X-Embodiment dataset, which spans multiple robotic platforms and embodiments, would require resources beyond the scope of this work. Nonetheless, Open X-Embodiment provides an ideal framework to assess cross-embodiment generalization, offering diverse manipulation and navigation demonstrations collected from real robots.

The purpose of this study is therefore not to achieve fully interactive control but to examine whether a medium-sized Vision-Language Model, such as LLaMA 3.2 Vision 11B, can be effectively adapted for robotic control through fine-tuning and prompting-based conditioning. Rather than designing a specialized control architecture, we treat the model as a general multimodal transformer and investigate to what extent it can learn to output 8-token control vectors when grounded on visual observations and textual instructions. This approach establishes a proof of

concept for using compact VLMs as scalable foundations for embodied intelligence.

At inference time, every predicted action is decoded into the 8-token format introduced in Chapter 5 (Section 5.3.1): one token for termination, three for Cartesian displacement, three for rotation, and one for gripper state, all obtained by discretizing continuous controls into 256 uniform bins mapped to reserved vocabulary IDs. Because the original LLaMA 3.2 Vision model is not inherently designed to output numeric control tokens, a dedicated conditioning phase was required.

Building upon this baseline, we then explored context-augmented fine-tuning, where the model receives a temporal window of four consecutive frames along with a short history of previously executed actions.

8.2 Evaluation Metrics

To assess model performance on Open X, we adopt a unified evaluation suite capturing discrete accuracy, continuous reconstruction error, and action-space alignment. These metrics are consistent with recent embodied learning frameworks and tailored to the 8-token action representation described above.

1. Per-sample Discrete Accuracy

We compute the fraction of correctly predicted tokens:

$$\text{TokenAcc} = \frac{1}{8}\sum_{i=0}^{7}\mathbf{1}[p_i = g_i],$$

and component-wise accuracies:

$$\text{PosAcc} = \frac{1}{3}\sum_{i=1}^{3}\mathbf{1}[p_i = g_i], \quad \text{RotAcc} = \frac{1}{3}\sum_{i=4}^{6}\mathbf{1}[p_i = g_i], \quad \text{GripAcc} = \mathbf{1}[p_7 = g_7].$$

The strict exact-match criterion is defined as

$$\text{ExactMatch} = \mathbf{1}[\forall i,\, p_i = g_i].$$

Discrete action tokens are detokenized into continuous control vectors $\hat{\mathbf{a}}$ and $\mathbf{a}_{\text{gt}}$, representing predicted and ground-truth actions, respectively. We evaluate reconstruction quality using absolute and normalized errors for position, rotation, and gripper components.

Absolute error. The absolute L2 distance between predicted and reference values quantifies the deviation in the model's continuous control prediction:

$$e_{\text{pos}} = \|\hat{\mathbf{v}}_{\text{pos}} - \mathbf{v}_{\text{pos}}^{\text{gt}}\|_2, \quad e_{\text{rot}} = \|\hat{\mathbf{v}}_{\text{rot}} - \mathbf{v}_{\text{rot}}^{\text{gt}}\|_2, \quad e_{\text{grip}} = |\hat{g} - g^{\text{gt}}|.$$

These quantities are expressed in native units (translation in normalized Cartesian space, rotation in radians, gripper in unitless scale).

Relative error. To make results comparable across components with different scales, we compute relative (normalized) errors by dividing each vector by its nominal span. Let s_{pos}, s_{rot}, and s_{grip} denote the valid dynamic ranges for translation, rotation, and gripper respectively (e.g., $s_{\text{pos}} = 2$ for range $[-1,1]$). The relative error is thus:

$$e_{\text{pos,rel}} = \frac{\|\hat{\mathbf{v}}_{\text{pos}} - \mathbf{v}_{\text{pos}}^{\text{gt}}\|_2}{s_{\text{pos}}}, \quad e_{\text{rot,rel}} = \frac{\|\hat{\mathbf{v}}_{\text{rot}} - \mathbf{v}_{\text{rot}}^{\text{gt}}\|_2}{s_{\text{rot}}}, \quad e_{\text{grip,rel}} = \frac{|\hat{g} - g^{\text{gt}}|}{s_{\text{grip}}}.$$

This normalization yields a dimensionless measure that reflects the fraction of the control range covered by the prediction error.

Per-axis RMS percentage error. Because position and rotation are three-dimensional, we report a more interpretable variant: the per-axis root-mean-square (RMS) percentage error obtained by averaging the normalized error over all axes:

$$\begin{aligned} e_{\text{pos,RMS\%}} &= \frac{e_{\text{pos,rel}}}{\sqrt{3}} \times 100, \\ e_{\text{rot,RMS\%}} &= \frac{e_{\text{rot,rel}}}{\sqrt{3}} \times 100, \\ e_{\text{grip,RMS\%}} &= e_{\text{grip,rel}} \times 100. \end{aligned}$$

This metric expresses the mean per-axis deviation as a percentage of the full dynamic range, providing a uniform scale to compare translation, rotation, and gripper performance. For example, a position RMS error of 2% indicates that, on average, each positional axis deviates by roughly 2% of its valid span.

Metric usage. In the following analyses, we report both the absolute and RMS percentage errors. Absolute L2 errors highlight the raw physical deviations, while the RMS percentage formulation enables fair cross-component comparison (position, rotation, and gripper) regardless of their original scales. This normalization is particularly useful when interpreting Open X metrics across embodiments and motion types, where the underlying spans differ in units and magnitude.

3. Action-Space Analysis

Beyond per-sample metrics, action-space measures assess how well the model aligns its per-step action deltas with the ground-truth commands over an entire episode. At each timestep we compare the predicted action vector (position delta, rotation delta, gripper command) with the corresponding ground-truth vector and compute directional scores, then aggregate these per-step values over time and across episodes. High scores therefore indicate that, on average, successive actions move in the intended direction and with roughly appropriate magnitude, while low scores reveal sequences that stall, oscillate, or drift away from the target motion. In our setting, the policy is trained and evaluated only on these per-step displacement vectors rather than on absolute end-effector poses, even though the underlying RLDS logs expose fields such as base pose of the end effector (Chapter 5). Classical path-tracking measures like along-track error are therefore not directly applicable, as they assume access to absolute poses and a continuous reference trajectory. Instead, we rely on local, delta-based criteria that evaluate whether the predicted deltas follow the correct direction, orientation, and relative progression of the ground-truth motion, rather than focusing solely on exact numerical values.

Directional Accuracy (DirAcc). In sequential robotic control, this notion of "progress" is crucial: a model may produce predictions that are close in absolute error yet fail to move the system in the correct direction, or conversely, may align directionally while underestimating magnitude. Directional Accuracy (DirAcc) measures, at each timestep i, how much of the ground-truth command the model applies along the correct direction, clipped to the range [0,1]. For a generic action component (position or rotation) with predicted vector $\mathbf{a}_{\text{pred}}(i)$ and ground-truth vector $\mathbf{a}_{\text{gt}}(i)$, we define

$$\text{DirAcc}(i) = \text{clip}\left(\frac{\mathbf{a}_{\text{pred}}(i) \cdot \mathbf{a}_{\text{gt}}(i)}{\|\mathbf{a}_{\text{gt}}(i)\|^2},\, 0,\, 1\right).$$

Here 1 means that the model matches or overshoots the ground-truth command in the correct direction, 0.5 corresponds to applying roughly half of the ground-truth magnitude along the right direction, and 0 indicates either no useful progress or motion in the opposite direction. The same definition is used for the gripper, treating its scalar command as a one-dimensional vector. DirAcc is computed independently for position, rotation, and gripper components. Typical interpretation thresholds are:

- Poor: < 0.2 - little or opposite progress.
- Acceptable: 0.3–0.5 - partial directional consistency.

- Strong: > 0.6 (excellent when ≥ 0.7) - accurate, magnitude-consistent motion.

This metric is not intended as a new standard, but rather as a task-specific diagnostic tool tailored to our robotic setting. It complements cosine similarity by penalizing both under- and over-shooting, and complements L2 error by evaluating alignment with the desired motion direction. Its role is to provide a sequence-level signal in action space indicating whether the model understands the structure of the task, even when discrete token predictions are imperfect.

Cosine Similarity (CosSim). To isolate angular agreement independently of step length, we compute, for the same pair $(\mathbf{a}_{\text{pred}}, \mathbf{a}_{\text{gt}})$:

$$\text{CosSim} = \frac{\mathbf{a}_{\text{pred}} \cdot \mathbf{a}_{\text{gt}}}{\|\mathbf{a}_{\text{pred}}\| \, \|\mathbf{a}_{\text{gt}}\|} \in [-1, 1].$$

CosSim reflects pure directional alignment: 1.0 indicates perfect orientation, 0 orthogonal movement, and -1.0 opposite direction. Interpretation thresholds are:

- Poor: < 0.2 or negative - divergent or reversed direction.
- Moderate: 0.3–0.5 - partial angular consistency.
- Good: > 0.6 (excellent when ≥ 0.7) - strong directional agreement.

While DirAcc measures "how far" progress advances toward the correct direction, CosSim measures "how aligned" the movement orientation is regardless of magnitude.

Angular Deviation (Angle Deg). We also express directional agreement in degrees for better interpretability:

$$\theta_{\text{deg}} = \arccos(\text{CosSim}) \times \frac{180}{\pi}.$$

Angles closer to 0° indicate better alignment ($< 45°$ very good, 60–90° moderate, $> 90°$ poor). This view makes it straightforward to assess how tightly the model follows the intended motion direction.

Aggregation across time and episodes. For both DirAcc and CosSim we report two types of aggregates. First, an episode-level score, obtained by averaging the per-step values within each episode and then taking the mean of these episode means across the test set; this emphasizes how consistently the controller behaves over complete trajectories. Second, a global step-weighted score, computed as the plain average over all timesteps of all episodes; this reflects the overall fraction of steps that are well aligned, regardless of how long each episode lasts.

8.3 Training Configurations and Loss Trends

We trained two models on the Open X-Embodiment dataset under distinct settings: a single-frame policy and a multi-frame policy. Both share the same LoRA configuration (rank 32, $\alpha = 32$, dropout=0.05), frozen vision and language backbones, and identical optimization setup (AdamW, cosine scheduler, mixed precision, ZeRO-2). The primary differences lie in temporal context, learning rate, and batch setup.

The single-frame configuration uses one RGB frame per step and predicts a single 8-token action vector. It was trained for five epochs with an effective batch size of 32 (16 samples, gradient accumulation 2) and a learning rate of 1×10^{-6}. This setup establishes the baseline reactive mapping between a static image and its corresponding control vector.

The multi-frame configuration, as we already mentioned, introduces a temporal window of four frames with 3 being the previous frames in the episode and the last being the current frame, enabling short-term reasoning over motion. Because this increases input dimensionality, the batch size was reduced to 4 (no gradient accumulation as we it noticeably degraded performances in the previous trainings on ALFRED) and the learning rate raised to 3×10^{-6} to maintain convergence speed. Additionally, an input dropout of 20% was applied to the action-history tokens, forcing the model to rely on visual cues when historical context is partially missing.

Figure 8.1 compares the loss curves of the two training regimes. The single-frame model exhibits a smooth and monotonic decay, stabilizing around 2.3 after 5,000 steps, indicative of consistent convergence. In contrast, the multi-frame model achieves a significantly lower final loss (around 1.8) but displays high-frequency oscillations throughout training. This instability is likely due to the increased temporal complexity and stochastic dropout on the previous-action tokens, which intermittently disrupts temporal consistency. Despite this, the overall downward trend suggests successful adaptation to the multi-frame temporal conditioning.

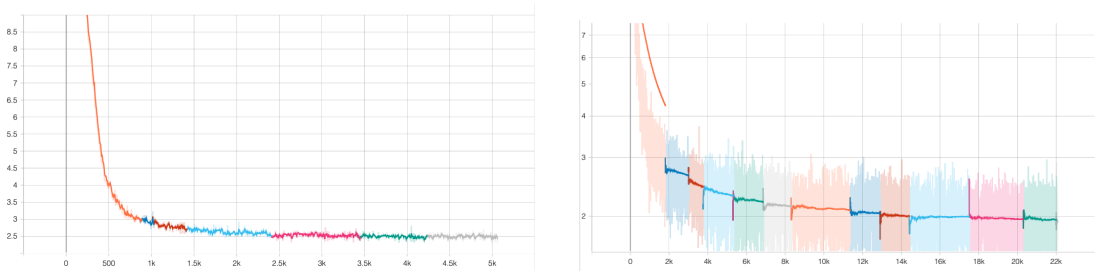


Figure 8.1: Loss trends for the single-frame (left) and multi-frame (right) training configurations. The multi-frame model achieves lower loss but exhibits higher variance due to temporal complexity and input dropout.

8.4 Evaluation and Results

The following section consolidates quantitative and qualitative results for the two policies considered in this study. We first present the single-frame baseline to establish reactive performance without temporal context, then analyse the multi-frame policy that incorporates short-term history and dynamic prompting. Finally, we provide a direct comparative analysis highlighting trade-offs, improvements, and limitations across both models.

Token Range, Quantization, and Label Imbalance Each control component is discretized into 256 uniform bins mapped to token IDs in the 126000–126255 range, with nominal spans of $[-1, +1]$ for translations and gripper motion and $[-\pi/2, +\pi/2]$ for rotations. This mapping simplifies interfacing with the language model but introduces representation effects that influence metrics and behavior.

First, since bins are spaced uniformly across the nominal span while most Open X-Embodiment actions lie in narrower intervals (approximately $[-0.6, +0.6]$ for position and about $[-1.3, +1.5]$ rad for rotation), high-density central regions are under-resolved: many fine-grained motion differences collapse into the same bin. Discretized errors can thus look smaller than the underlying continuous deviations. See the distribution plots in the dataset chapter (Figures 5.4, 5.5, 5.6).

Second, the relative errors we report, namely the per-axis RMS percentage errors for position and for rotation, are normalized using the nominal spans, not the actual ranges present in the data. Because the robot typically moves within a smaller range than nominal, the same absolute error takes up a larger share of the true range. In practice this means percentages like 1.5–2.4% are optimistic: if we rescale by the empirical data ranges (or by the RT-1 control ranges), the percentage errors become higher.

In addition, the label distribution is strongly imbalanced. Central token values around 127–128 for translations and gripper, and the termination token near 1, dominate the dataset, as shown by the single-frame token histograms (Figures 5.5, 5.6) and the multi-frame panels (Figures 5.8, 5.9). In the single-frame setting, limited context encourages regression toward these frequent tokens and makes it difficult to detect episode boundaries. In the multi-frame setting, temporal context and prompting reduce reliance on priors and produce more coherent, context-aware action sequences despite the imbalance.

8.4.1 Single-Frame Policy

The single-frame policy serves as a foundational approach for mapping visual observations to robot control actions. In this configuration, the model receives a single RGB frame and is tasked with predicting the corresponding 8-token

control vector for each timestep. This setup is inherently reactive, relying solely on instantaneous spatial cues without access to temporal information from previous or future frames. As such, while it can capture immediate visual context, it may face challenges in producing stable and coherent sequences of actions across consecutive timesteps. The evaluation was conducted on a total of 8,094 samples drawn from the Open X-Embodiment dataset, corresponding to multiple episodes across diverse manipulation tasks.

Prompting Strategy For both training and evaluation, the model is conditioned using a templated prompt of the form:

<image> What should the robot do to '{instruction}'?

This prompt structure introduces the rendered frame and directly links the provided natural-language instruction to the desired action prediction. The single-turn prompting approach maintains a lightweight interface, explicitly grounding the model's attention toward the task objective specified by the instruction. This represents a minimal grounding strategy that will later evolve into more structured and context-rich prompting techniques.

Grid Search and Inference Parameter Analysis To establish a robust inference protocol, we conducted a grid search over key decoding hyperparameters. The parameters explored include temperature (which sharpens or flattens the output distribution), top-p (nucleus sampling, restricting sampling to the most probable tokens whose cumulative probability exceeds a threshold), top-k (restricting sampling to the top k tokens), and the do_sample flag (which toggles between stochastic sampling and deterministic greedy decoding). When do_sample is set to False, decoding becomes deterministic, rendering top-p and top-k inactive, while temperature continues to affect the sharpness of the output probabilities.

The results of this hyperparameter sweep are summarized in Table 8.1. The highest exact-match accuracy achieved is approximately 0.0039 (0.39% of actions with all eight tokens correct), which is consistent with the stringent nature of the exact-match criterion. Consequently, the primary focus at this stage is not on achieving perfect action sequences but on ensuring that the model reliably predicts the correct control direction. For subsequent analyses, we adopt the configuration with greedy decoding as the reference setting.

Tag	Temperature	Top-p	Top-k	do_sample	Seq. acc. ($\times 10^{-3}$)	Token acc. (%)
G0	N/A	N/A	N/A	False	3.94	28.87
G1	0.2	0.9	0	True	3.33	28.95
G3	0.7	0.95	0	True	2.46	26.09
G2	0.5	0.92	40	True	2.34	28.02
G4	0.9	0.8	50	True	0.99	25.86

Table 8.1: Grid search over decoding parameters for the single-frame policy. Sequence accuracy corresponds to the fraction of actions whose eight tokens exactly match the ground truth; values are expressed as $\times 10^{-3}$ to highlight the low exact-match rate. Token accuracy reports the average percentage of correctly predicted tokens.

Quantitative Evaluation

We summarize the primary metrics for the single-frame policy in Table 8.2. Results confirm that the discrete objective is not learned in a task-consistent way: exact-match remains below 0.39%, while mean per-token accuracy is 28.87%. Absolute and span-normalized (relative) errors appear small, but, as discussed later, this is largely an artifact of the tokenization span and value concentration around zero.

Metric	**Value**
Exact match (%)	0.39
Token accuracy (mean, %)	28.87
Position accuracy (discrete, %)	19.33
Rotation accuracy (discrete, %)	15.60
Gripper accuracy (discrete, %)	31.13
Position error (abs, unitless)	0.0825
Rotation error (abs, rad)	0.1791
Gripper error (abs, unitless)	0.6461
Position error (RMS/axis, %)	2.38
Rotation error (RMS/axis, %)	3.29
Gripper error (relative, %)	32.30
Terminate recall / precision (%)	0.0 / 0.0
Positive / negative supports	402 / 7713

Table 8.2: Primary quantitative metrics for the single-frame policy evaluated on 8,094 samples. Percentages report means multiplied by 100.

Discrete accuracy. Discrete metrics are markedly low: the exact-match rate does not exceed 0.39%, indicating that the model rarely produces fully correct 8-token actions. Qualitative inspection shows frequent collapses toward high-prior patterns such as "1 128 128 128 128 128 128 128", the statistically most probable token tuple rather than task-consistent actions. While exact-match is a strict criterion for discretized continuous control, even per-component accuracies remain modest (position 19.33%, rotation 15.60%, gripper 31.13%). For this reason, we refrain from presenting figures in this subsection and focus on the numerical evidence.

Continuous Error Analysis. Absolute and relative measures offer complementary views of continuous reconstruction. In absolute terms, mean positional deviation is 0.0825 (normalized Cartesian units) and mean rotational deviation is 0.179 radians. To compare heterogeneous dimensions, errors are also expressed as a fraction of the nominal control span ($[-1, +1]$ for translation and gripper,

$[-\pi/2, +\pi/2]$ for rotation). We report both the global relative error and the per-axis RMS percentage, the latter dividing the 3-D relative error by $\sqrt{3}$ and multiplying by 100 to reflect the average deviation on each axis. Under this convention, position and rotation attain 2.38% and 3.29% RMS per-axis (about 4.1% and 5.7% on the non-RMS vector), respectively.

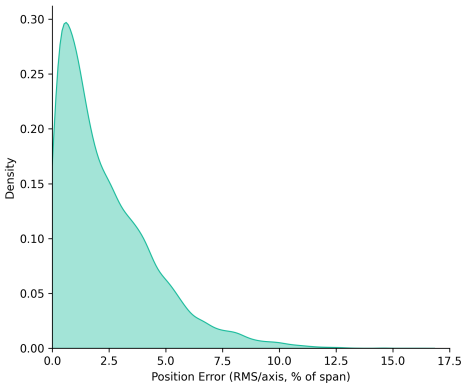
As discussed earlier in the span analysis, these percentages are conservative because normalization uses nominal rather than empirical ranges. Furthermore, the apparent low error values can be partly attributed to the model's tendency to exploit the statistical bias of the dataset. In practice, by frequently predicting the central token (around 127), which corresponds to near-zero deltas in both position and rotation, the model minimizes reconstruction loss even without learning meaningful control dynamics. This behavior is reinforced by the fact that most ground-truth deltas for translation and rotation are themselves concentrated near zero, making such "static" predictions yield deceptively low continuous errors despite producing a dummy motion.

Figure 8.2 shows the distributions for position and rotation. Both are sharply peaked near zero with heavier tails for rotation, indicating that rotational components are harder to reconstruct. Most mass lies between 2–4% RMS, but the tails extend toward $\sim$10%, consistent with occasional drift over time when operating without temporal context.

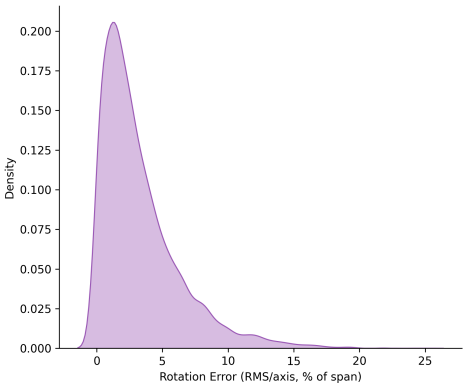
The gripper exhibits a distinct pattern. Because its control is effectively quasi-binary (open vs close, with short transients), the error distribution is bimodal and much broader, yielding a mean relative error around 32%. This reflects two effects highlighted in the token-imbalance discussion: (i) a strong prior toward the most frequent "open" token and (ii) the fact that gripper uses the full $[-1, +1]$ span more often than translation and rotation deltas concentrated near zero. Consequently, small absolute mistakes translate into larger relative percentages for the gripper than for position or rotation.

Quantization plays a minor role here. With 256 bins, the expected per-axis quantization noise is roughly 0.23% of span for translation and 0.35% for rotation, well below the empirical errors; the limiting factor is model prediction, not discretization. This is corroborated by Figure 8.3, where absolute position errors versus detokenized-discrete and original-continuous targets are nearly overlapping.

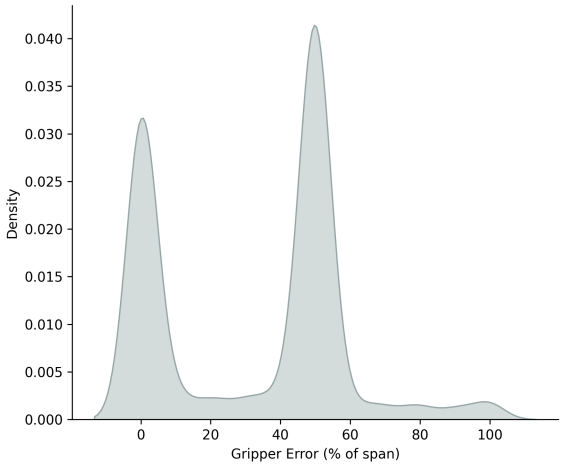
Efficiency and memory footprint. As discussed in Chapter 6, loading the base LLaMA 3.2 Vision checkpoint already consumes roughly 23 183 MiB on the A100 80GB GPU used in our experiments. In the Open X single-frame configuration, peak inference memory remains close to this baseline (approximately 23 044 MiB), and the mean inference time is around 0.63 s per sample, corresponding to a control frequency of roughly 1.6 Hz. While this latency appears modest in absolute terms, it is far below the tens of Hertz typically required for closed-loop robot control,



Position error (per-axis RMS, percent of span).



Rotation error (per-axis RMS, percent of span).



Gripper error (relative, percent of span).

Figure 8.2: Single-frame baseline: relative error distributions for position, rotation, and gripper.

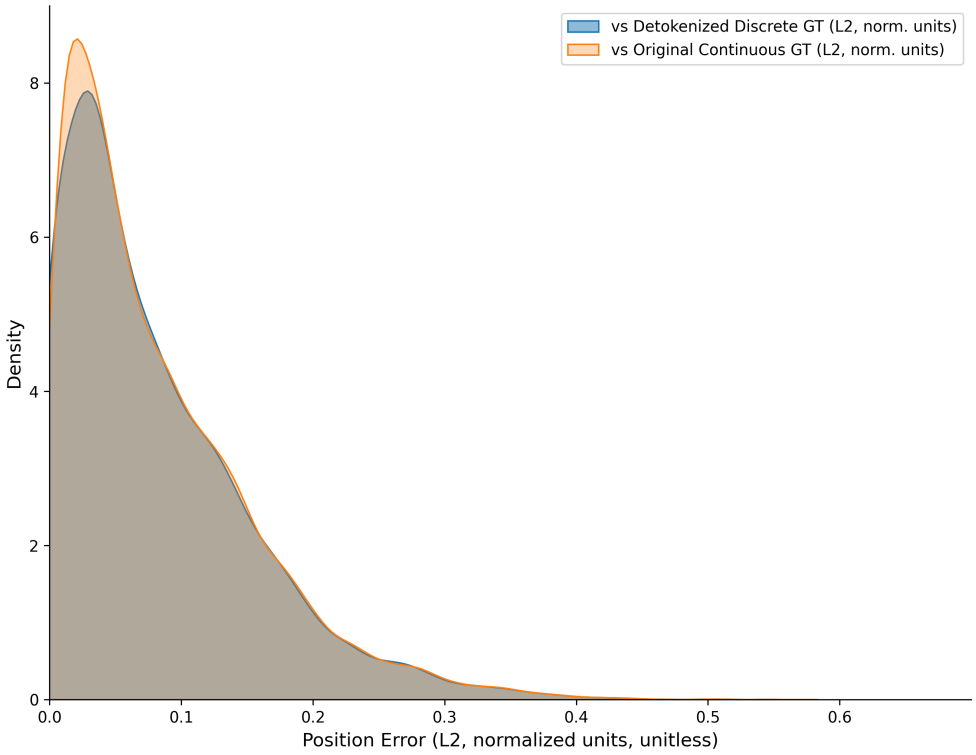


Figure 8.3: Absolute position error (L2, normalized units) against detokenized-discrete and original-continuous targets. The curves are nearly overlapping, indicating limited impact of quantization.

confirming that the single-frame configuration is suitable for offline analysis rather than real-time deployment.

Terminate token failure. The first token in the 8-token action vector encodes whether the episode should terminate (positive class) or continue (negative class). Under the current class mapping, the model never predicts the termination class, yielding 0.0% recall and 0.0% precision on positives (support 402). In other words, it always outputs the non-terminating value, so episodes are never explicitly closed by the learned policy. The resulting confusion counts are reported in Table 8.3. For this diagnostic the key quantity is therefore the number of predicted positives, which remains identically zero. This behavior points to a strong prior and/or inadequate loss weighting for the class-imbalanced termination token; class-balanced or focal losses for the first token are advisable.

	TP	FP	TN	FN
Terminate detection	0	0	7692	402

Table 8.3: Confusion counts for the termination detector. Positives correspond to terminating timesteps, negatives to non-terminating ones. The model never predicts the positive class: all ground-truth terminations are missed (FN=402), with no false positives on non-terminating steps (TN=7692).

Action-Space Analysis. Action-space metrics based on DirAcc and CosSim summarise how well, on average, the per-step action deltas predicted by the model align with the corresponding ground-truth commands across an episode. For the single-frame policy, these metrics reveal that the controller behaves as a purely reactive system, with spatial actions that are essentially misaligned with the reference commands.

Quantitatively, the mean directional accuracies remain extremely low for position and rotation (about 0.06 and 0.03 respectively), while the gripper achieves a moderate value around 0.5 depending on the aggregation scheme (episode mean versus global step-weighted mean). Cosine similarities are even more discouraging for the spatial components: position and rotation average roughly -0.05 and -0.09, corresponding to angles near 93° and 95°, that is, motion that is slightly biased toward the wrong direction rather than aligned with the target.

The per-episode histograms (Figure 8.6) make this pattern explicit. Most episodes cluster near zero directional accuracy for position and rotation, confirming that translational and rotational commands rarely advance along the correct direction. In contrast, the gripper distribution is centered around 0.5, indicating

that the model often applies roughly half of the desired opening or closing command in the right direction, even though this scalar consistency does not translate into coherent spatial motion.

Step-wise aggregate plots (Figures 8.4–8.5) further confirm the absence of structured behavior in the spatial channels. For each step index we average DirAcc and CosSim across all episodes that reach that step, so the curves summarize typical alignment for actions occurring at the same relative position in the sequence rather than tracking a single trajectory. The mean DirAcc curves for position and rotation hover close to zero throughout and exhibit erratic fluctuations, while the gripper curve oscillates around 0.5 for the first 70–80 steps before becoming dominated by noise when only a few long episodes remain. Cosine similarities for position and rotation stay mildly negative over most of the horizon and become increasingly unstable toward the end, reflecting both cumulative drift and the small number of trajectories contributing at high step indices.

Taken together, these results indicate that the single-frame model does not exhibit true trajectory-following capability. It tends to produce quasi-static spatial outputs (often the most frequent central tokens), achieving deceptively low continuous errors while failing to move along the ground-truth motion directions. In practical terms, the policy performs what can be described as dumb inference: it minimizes the loss function through statistical bias rather than by learning physically consistent motion patterns.

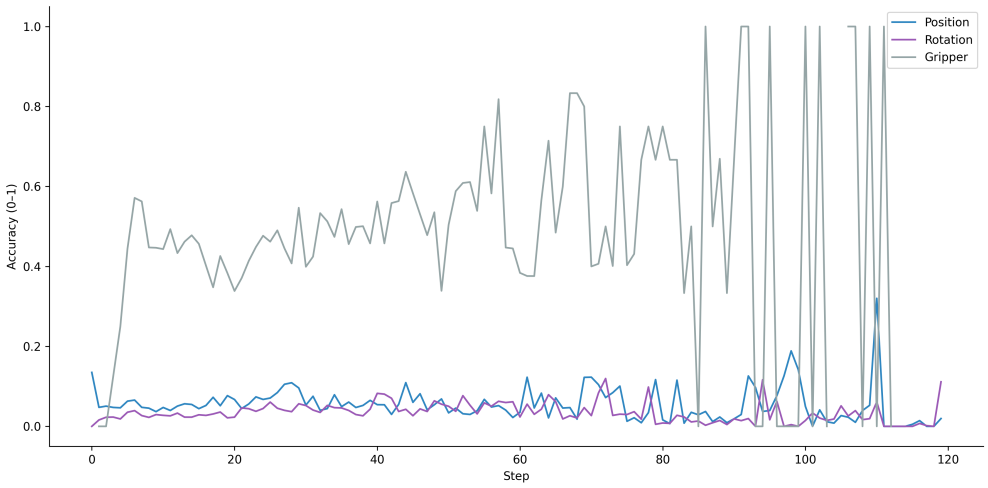


Figure 8.4: Mean directional accuracy over steps for position, rotation, and gripper. Position and rotation remain close to zero at all horizons, while the gripper hovers around 0.5 for early steps and then becomes noisy once only a few long episodes contribute, confirming that spatial motion lacks consistent progression even when gripper commands are partially aligned.

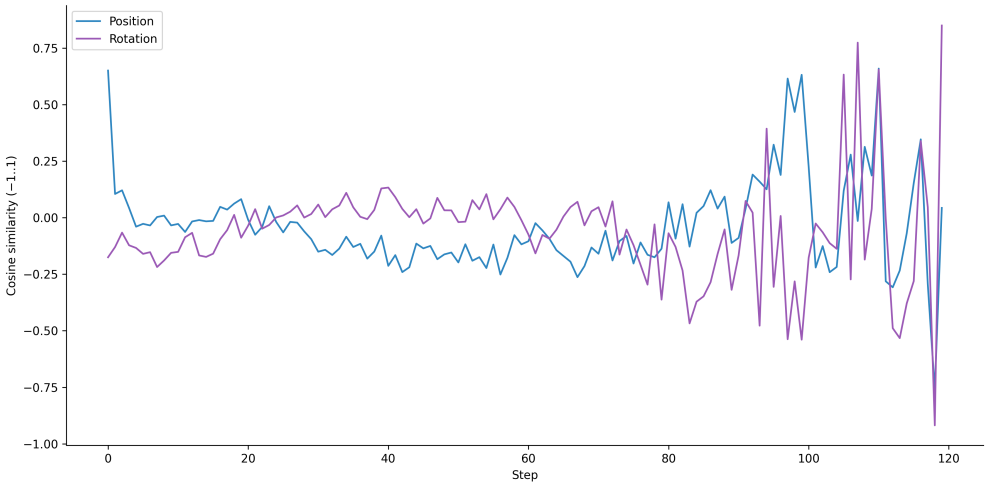
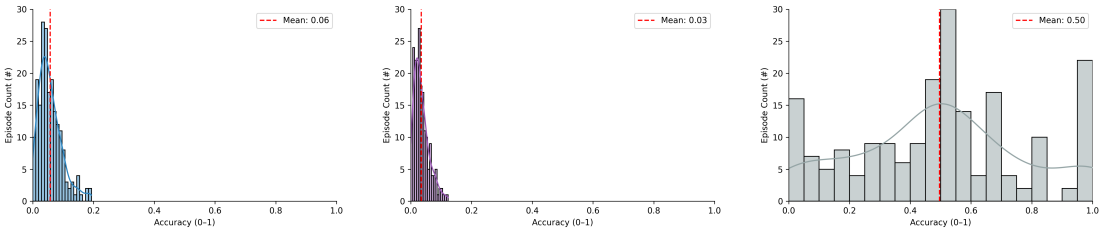


Figure 8.5: Mean cosine similarity over steps for position and rotation. Values stay mildly negative over most of the episode and become highly unstable toward the end, indicating that predicted spatial actions are often oriented opposite to the ground-truth motion rather than aligned with it.



Position directional accuracy distribution. Mean: 0.04.

Rotation directional accuracy distribution. Mean: 0.02.

Gripper directional accuracy distribution. Mean: ≈ 0.5.

Figure 8.6: Episode-level directional accuracy distributions for position, rotation, and gripper. Position and rotation are strongly concentrated near zero, confirming the lack of meaningful spatial alignment, whereas the gripper is centered around 0.5, indicating partial but not full alignment of opening and closing commands.

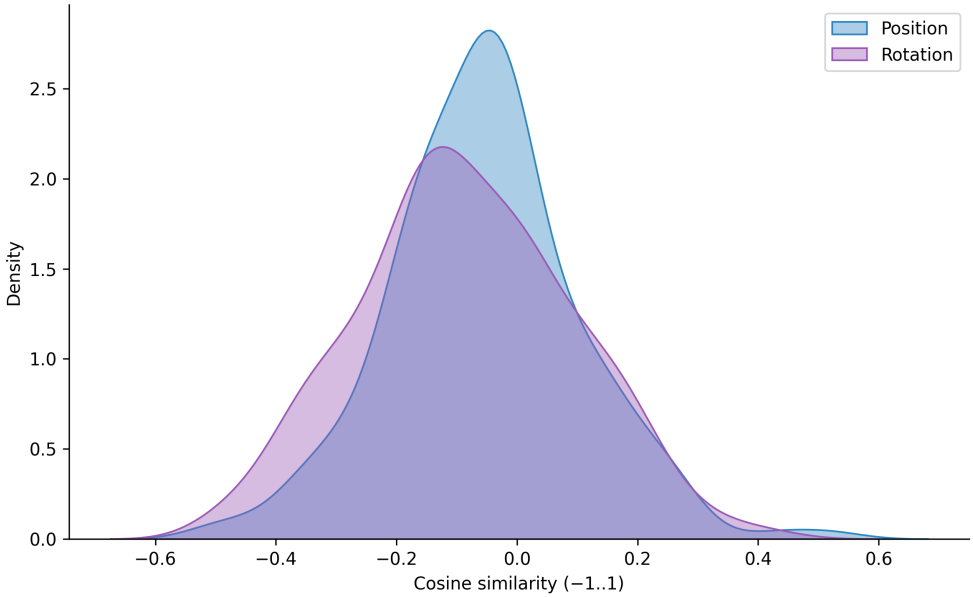


Figure 8.7: Episode-level cosine similarity distributions for position and rotation. Both distributions are centered slightly below zero, indicating a systematic bias toward misaligned or even reversed motion directions rather than genuine alignment with the ground truth.

Despite these limitations, the single-frame policy serves as a valuable diagnostic baseline. It reveals the constraints of purely reactive vision-to-action mapping and highlights how statistical biases dominate when temporal or contextual cues are absent. The findings underscore the necessity of richer multimodal grounding to achieve temporally coherent control.

In the following section, we extend this framework to a multi-frame policy trained on a larger subset of Open X data. This new configuration introduces both a more explicit prompting strategy and a temporal observation window that includes the most recent frames and previously executed actions. The goal is to assess whether short-term temporal conditioning and contextual grounding can mitigate the limitations observed in the single-frame setup, improving both action-space alignment and control stability.

8.4.2 Multi-Frame Policy

The multi-frame policy extends the single-frame baseline by introducing temporal conditioning, a richer dataset and a more structured prompting strategy, allowing us to assess whether short-term temporal context improves the coherence and causal consistency of generated control sequences. The experiment is conducted on a larger and more diverse subset of the Open X-Embodiment dataset comprising roughly 2,300 unique episodes and 127,381 total samples, split into 110,752 for training (86.9%) and 16,629 for testing (13.1%). The test split covers 300 unique episodes, each with consecutive RGB observations and their corresponding low-level robot actions across a variety of real-world manipulation tasks.

A temporal observation window of four consecutive frames, chosen as a practical hyperparameter, provides short-term motion context while maintaining computational efficiency, and the model is fine-tuned for a single epoch on this extended dataset following the setup described in Section 8.3. Compared to the single-frame policy, this configuration trades longer training duration for broader data diversity and explicit temporal reasoning, aiming to strengthen the consistency and causality of generated control trajectories. As part of this study we evaluated the multi-frame policy under two inference regimes to assess its dependence on temporal context: in the primary configuration, Full-Context Inference, all previous observations and action-history tokens are provided to the model, corresponding to the intended deployment setting where full temporal information is available, whereas in Masked-Context Inference an input-dropout probability of 0.2 is applied at inference time so that portions of the history tokens are randomly removed, simulating situations where parts of the recent action sequence are unavailable or ambiguous. This comparison quantifies how strongly the model relies on explicit temporal conditioning and whether it can still infer motion direction and intent from visual cues alone.

Prompt Design and Temporal Grounding. In this setting, grounding denotes the process of establishing consistent links between language, visual observations, and low-level actions. At the perceptual level, words in the instruction and prompt are expected to anchor to specific regions in the images, such as the robot gripper, the manipulated object, or the target surface. At the action level, the eight control tokens must correspond to observable changes in the robot state, so that a predicted translation, rotation, or gripper command is reflected in the subsequent frame. Temporal grounding further requires each prediction to be conditioned on the history of observations and previously executed actions, enabling the model to reason about an evolving trajectory rather than a single frame in isolation.

As observed in the ALFRED experiments presented in previous chapters, the textual prompt plays a crucial role in aligning visual and linguistic modalities. For this policy, the prompt was redesigned to explicitly encode temporal order and causal relationships between past observations, executed actions, and the current visual state.

Each training example follows a structured prompt composed of three main stages:

1. **Temporal framing.** The input begins with a sequence of past observations and corresponding actions, denoted as:

   Observation 1: [image] → Robot performed action: [tokenized action]

   This is repeated for up to three previous timesteps. This structure provides the model with a notion of temporal flow and causality, namely what the robot has already seen and done.

2. **Grounding.** After the temporal context, a concise grounding statement follows:

   Focus on the robot arm pose, gripper status, and objects on the table.

   This phrase acts as a visual attention anchor, orienting the cross-modal alignment toward key elements of the scene.

3. **Goal-conditioned reasoning.** Finally, the current frame is introduced with the instruction-conditioned query:

   Based on the observation history and the current observation, what should the robot arm do to [instruction]?

   This stage explicitly links the temporal history with the task objective, prompting the model to infer the next appropriate control action.

An input dropout of 0.2 was applied to the action-history tokens during training, randomly removing the tokenized representations of previous actions from the prompt and forcing the model to rely more heavily on visual cues when temporal information is incomplete. In practice this serves two purposes: it prevents overfitting to fixed action patterns in the training data and it regularizes cross-modal attention by encouraging the model to infer missing context from the visual sequence rather than memorizing explicit action histories.

This prompting scheme transforms the task from a static image-to-action mapping into a temporally grounded reasoning process, and, as the cross-attention maps will later show, visual tokens from past frames are effectively linked to their corresponding action tokens, indicating that the model learns to attend selectively to historical visual evidence when generating its next control output.

Quantitative Evaluation

We now report the quantitative results obtained with the multi-frame policy. The evaluation was performed on the 16,629 sample test split (300 unique episodes) using the same metric suite described in Section 8.4. Table 8.4 summarizes the main results.

Metric	**Value**
Exact match (%)	3.41
Token accuracy (mean, %)	38.68
Position accuracy (discrete, %)	24.62
Rotation accuracy (discrete, %)	19.97
Gripper accuracy (discrete, %)	77.39
Position error (abs, unitless)	0.0517
Rotation error (abs, rad)	0.1159
Gripper error (abs, unitless)	0.0999
Position error (RMS/axis, %)	1.49
Rotation error (RMS/axis, %)	2.13
Gripper error (relative, %)	4.99

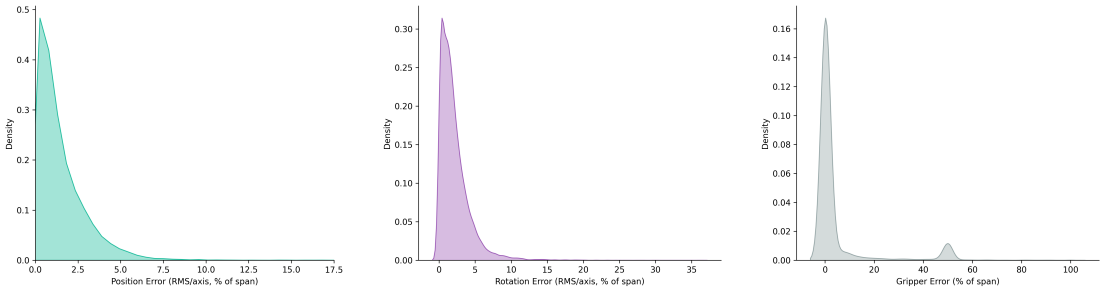
Table 8.4: Primary quantitative metrics for the multi-frame policy evaluated on 16,629 samples. All values represent mean percentages over the test set.

The improvements over the single-frame policy are substantial across all dimensions. Exact-match accuracy increases by nearly an order of magnitude (from 0.39% to 3.4%), and mean token accuracy rises from 28.9% to 38.7%. Among the control components, the gripper benefits most, reaching 77.4% discrete accuracy, more than double the previous score. This reflects the model's ability to recognize

the opening/closing phases typical of manipulation episodes, rather than simply outputting the most frequent token. Position and rotation accuracies also improve significantly (roughly +5–8%), though their relative improvement appears moderate due to the compressed span of valid motion values discussed earlier.

Continuous Error Analysis. In absolute terms, the mean positional deviation decreases to 0.0517 (normalized units), and the mean rotational deviation to 0.116 rad. Expressed as RMS-per-axis fractions of span, these correspond to 1.49% for position and 2.13% for rotation, almost a twofold reduction relative to the single-frame baseline (2.38% and 3.29%). The gripper shows an even greater improvement, with its relative error dropping from 32.3% to only 5.0%. These results indicate that temporal grounding and richer contextual prompts enable much more stable reconstruction of continuous actions.

Figure 8.8 visualizes the relative error distributions for position, rotation, and gripper. Compared to the single-frame case, the distributions are notably narrower and more peaked near zero, with the bulk of samples lying within 0–5% of the nominal span. The gripper's distribution remains bimodal but the secondary lobe around 50% of span, corresponding to mispredicted opening/closing transitions, is now greatly attenuated, indicating improved phase discrimination.



Position error (RMS/axis, percent of span).

Rotation error (RMS/axis, percent of span).

Gripper error (relative, percent of span).

Figure 8.8: Multi-frame policy: relative error distributions for position, rotation, and gripper. All curves are more concentrated near zero compared to the single-frame baseline, confirming improved reconstruction fidelity.

As with the single-frame case, quantization effects remain negligible. Figure 8.9 compares the L2 position errors computed against both the detokenized discrete and original continuous ground-truth vectors. The near-overlapping curves confirm that discretization noise, theoretically below 0.3% of the span, is well within measurement noise and model variability.

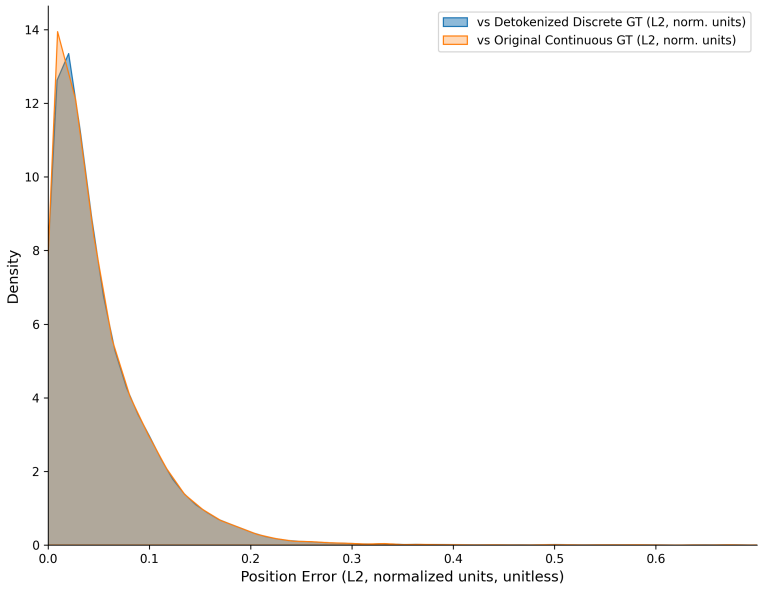


Figure 8.9: Absolute position error (L2, normalized units) against detokenized-discrete and original-continuous ground truth. The overlap of the two curves indicates that quantization noise is negligible compared to model prediction error.

Termination detection. A particularly notable improvement emerges in the termination flag prediction. As in the single-frame case, the first token represents a binary decision where positives correspond to terminating timesteps and negatives to non-terminating ones. Each episode contributes two terminating steps in the labels, but the model usually predicts only the final one (with about 20 episodes as exceptions). Despite this imbalance, the multi-frame policy reaches 99.4% precision and 53.3% recall on the positive class, corresponding to an F1 score of 0.69. In absolute terms, it recovers 320 ground-truth terminations with 2 false alarms and 280 misses (Table 8.5), marking a clear improvement over the single-frame baseline where the termination class was never predicted. This behavior indicates that temporal conditioning allows the model to recognize episode endings with high confidence once sufficient contextual evidence accumulates.

	TP	FP	TN	FN
Terminate detection	320	2	16,027	280

Table 8.5: Confusion counts for termination detection under the multi-frame policy. Positives correspond to terminating timesteps, negatives to non-terminating ones. The model correctly identifies over half of the true episode endings.

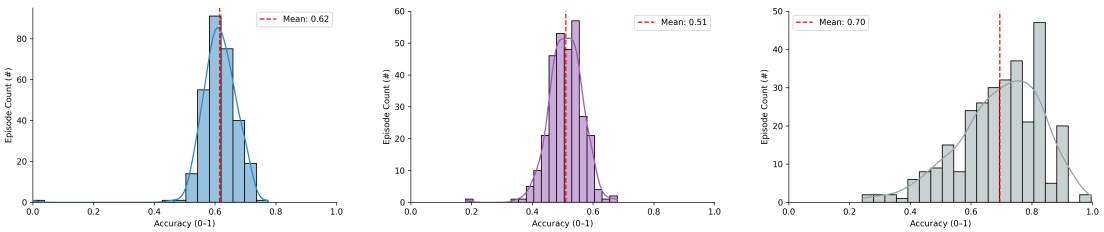
Thus, these results highlight the effectiveness of temporal conditioning. The model no longer collapses toward static, high-prior token patterns but instead learns to modulate its outputs consistently with recent visual and action history. Although the numerical differences in RMS error appear small in absolute terms, within the compressed operational range of Open X they correspond to meaningful gains in control precision and phase recognition.

Efficiency and memory footprint. From a systems perspective, the multi-frame policy is substantially heavier than the single-frame baseline. Instantiating the base LLaMA 3.2 Vision checkpoint in the Open X setup already consumes roughly 23 183 MiB on the 80 GiB GPU, even before any images are processed. During inference with a four-frame window and the extended temporal prompt, peak memory increases to about 28 636 MiB, and the mean inference time rises to approximately 1.88 s per sample, corresponding to a control frequency of only ~ 0.53 Hz. This slowdown reflects both the larger visual context and the longer textual conditioning required by the multi-frame prompt. As in the single-frame case, these numbers are acceptable for offline evaluation but fall well below the frequencies typically needed for responsive real-world control, motivating more compact models or aggressively quantised deployments.

Action-Space Analysis. Compared with the single-frame baseline, where spatial DirAcc scores were close to zero and cosine similarities hovered around or below zero, the multi-frame configuration exhibits a clear and substantial improvement on all components. Using the episode-level aggregates, the mean DirAcc reaches approximately 0.62 for position, 0.51 for rotation, and 0.70 for the gripper, while the corresponding cosine similarities average about 0.82 for position and 0.73 for rotation, with angles around 35° and 43°. In the taxonomy introduced earlier, position and gripper operate firmly in the "strong" regime, and rotation lies at the upper end of the "acceptable" band.

These results indicate that the multi-frame controller is generally able to steer its predicted deltas along the correct direction with a reasonably accurate magnitude: most steps now advance along the ground-truth motion instead of drifting randomly or reversing direction, and a non-trivial fraction of episodes achieve almost perfect directional alignment. At the same time, the metrics still fall short of near-perfect per-step agreement (for example DirAcc and CosSim close to 1), so individual actions remain somewhat noisy and occasionally over- or under-shoot the desired command, especially at later timesteps where error accumulation and sparse statistics play a larger role.

The per-episode histograms in Figure 8.10 illustrate this improvement clearly: the distributions are centered around relatively high mean values (about 0.62, 0.51, and 0.70), and most mass lies above 0.5, indicating strong directional alignment across episodes. The rotation distribution remains slightly broader than that of position, confirming that rotational control continues to be more challenging, while the gripper shows the most stable and peaked profile, consistent with its simpler scalar nature.



Position directional accuracy distribution. Mean: 0.62.

Rotation directional accuracy distribution. Mean: 0.51.

Gripper directional accuracy distribution. Mean: 0.70.

Figure 8.10: Episode-level directional accuracy distributions for position, rotation, and gripper. Compared to the single-frame policy, most episodes now exhibit strong directional alignment with the ground-truth trajectories, with position, rotation, and gripper commands concentrated well above 0.5.

The cosine similarity distributions (Figure 8.11) further confirm the trend with both position and rotation curves have shifted decisively toward high positive cosine values, with most mass between 0.6 and 0.9 and a subset of samples approaching 1.0 for position and about 0.9 for rotation. This transition from a nearly symmetric distribution around zero (in the single-frame case) to a tight right-skewed peak demonstrates that the model now predicts motion directions largely consistent with the underlying dynamics.

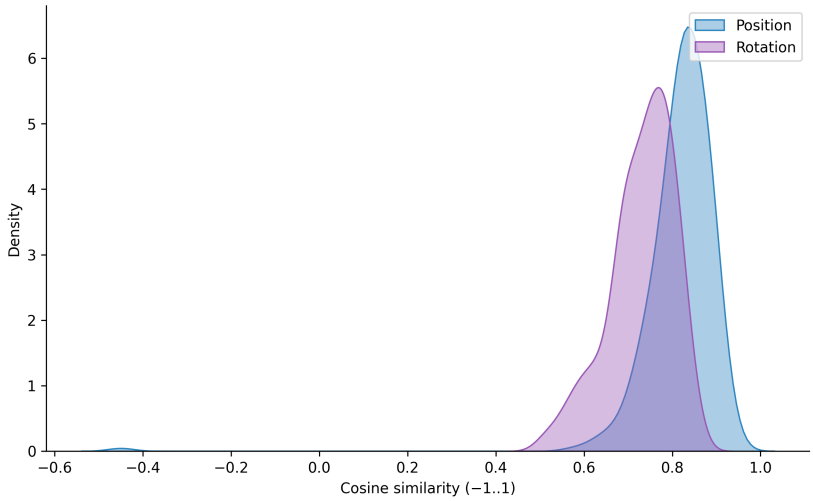


Figure 8.11: Episode-level cosine similarity distributions for position and rotation. Both curves are tightly concentrated between 0.6 and 0.9, with the position density reaching values close to 1.0 and rotation peaking around 0.9, indicating that many episodes achieve near-perfect directional alignment in action space.

Step-wise aggregate plots (Figures 8.12 and 8.13) provide additional insight into how these metrics behave at different step indices. At each index we average DirAcc and CosSim across all episodes that contain that step, so the curves summarize typical alignment for actions occurring early, mid, or late in the sequence rather than the evolution of any single trajectory. The mean DirAcc stabilizes around 0.6 for position and 0.5 for rotation over most of the horizon, while cosine similarities remain high. The sharp oscillations that appear after roughly 60 steps have two main causes. Statistically, only a handful of long episodes contribute to the tail of the curves, so each late timestep averages over very few samples and can be dominated by a single atypical trajectory. From a modeling perspective, error accumulation at long horizons makes the remaining trajectories more unstable: some policies keep making reasonable corrections while others overshoot or jitter, and with so few episodes this mixture shows up directly as large swings in DirAcc and cosine similarity.

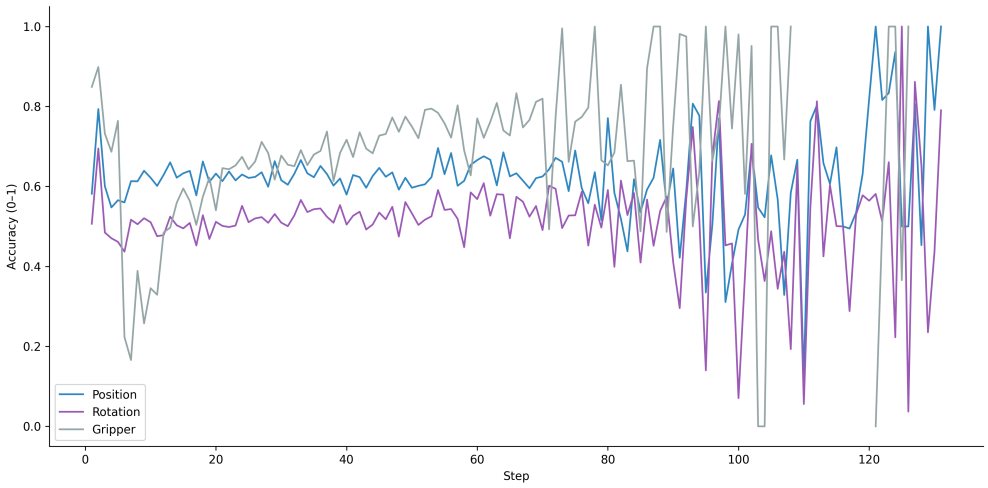


Figure 8.12: Mean directional accuracy per step for position, rotation, and gripper. Position and rotation stabilize around 0.6 and 0.5 for most of the horizon, while the gripper climbs from low values in the initial steps to peaks close to 1.0 before the tail becomes noisy when only a few long episodes remain, confirming strong but not perfectly stable forward progress.

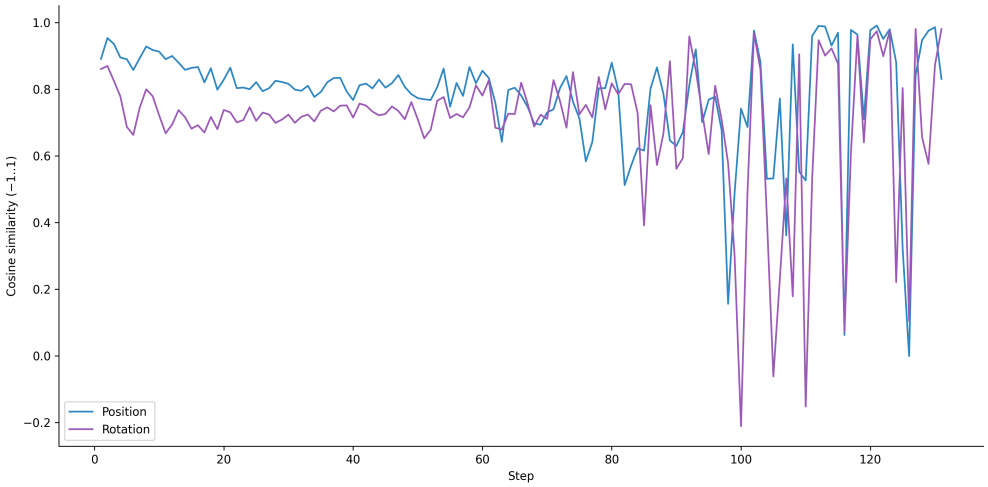


Figure 8.13: Mean cosine similarity per step for position and rotation. Cosine values remain high throughout most of the trajectory (around 0.8 for position and 0.7 for rotation), with sharp drops only in the final steps where very few episodes contribute, indicating robust directional alignment whenever sufficient temporal context is available.

Eval Context	Token Acc	Exact	Pos RMS	Rot RMS	DirAcc (P,R,G)	Cos (P,R)
Full	38.68	3.41	1.49	2.13	0.62/0.51/0.70	0.82/0.73
Masked (p=0.2)	36.69	2.64	2.19	2.88	0.55/0.43/0.70	0.73/0.59

Table 8.6: Ablation on history-token dropout. Models are trained with input-dropout p=0.2 and evaluated either with full temporal context or with masked previous actions. Percentages for Token Acc, Exact, and RMS columns.

While the results are not yet sufficient for real-world deployment, the improvement over the single-frame baseline is substantial: the model no longer performs random or prior-driven inference but instead generates directionally coherent motion aligned with sequence history. It effectively learns short-term kinematic consistency, laying the foundation for future extensions toward closed-loop embodied control. It is worth noting that these evaluations were conducted without input dropout at inference time; as shown in subsequent experiments, reintroducing a 0.2 dropout on previous actions during evaluation leads to degraded performance, confirming that the strongest improvements rely on leveraging complete temporal context.

Ablation: Effect of History-Token Dropout. To assess the model's dependence on explicit temporal context, we conducted an ablation study by re-evaluating the multi-frame policy with an input-dropout probability of 0.2 applied at inference time. This configuration (Masked-Context Inference) randomly removes previously tokenized actions from the prompt, mimicking situations in which the robot receives incomplete or noisy temporal information. All other evaluation parameters remained identical to the Full-Context run.

As summarized in Table 8.6, masking historical context at inference time consistently degrades performance. Mean token accuracy drops from 38.7 % to 36.7 %, and exact-match decreases from 3.4 % to 2.6 %. Continuous reconstruction errors roughly double in relative terms (position RMS from 1.49 % $\rightarrow$ 2.19 %; rotation RMS from 2.13 % $\rightarrow$ 2.88 %), confirming that the model relies heavily on temporal cues to maintain geometric precision.

Action-space metrics are particularly affected. Directional accuracies fall from about 0.62 to 0.55 for position and from 0.51 to 0.43 for rotation, and cosine similarities drop from roughly 0.82 to 0.73 (position) and from 0.73 to 0.59 (rotation), moving away from the strong-alignment regime toward more moderate agreement. Interestingly, the gripper DirAcc remains almost unchanged (about 0.70 in both settings), indicating that the main degradation concerns spatial alignment rather than open/close commands.

Figures 8.14 and 8.15 illustrate these trends. When history information is masked, the per-step DirAcc curves for position and rotation settle around lower plateaus

and exhibit stronger fluctuations, while the gripper retains roughly the same rising profile observed under full context. Cosine-similarity plots and distributions remain clearly positive but shift left and become broader, indicating weaker and less stable directional alignment between predicted and ground-truth actions.

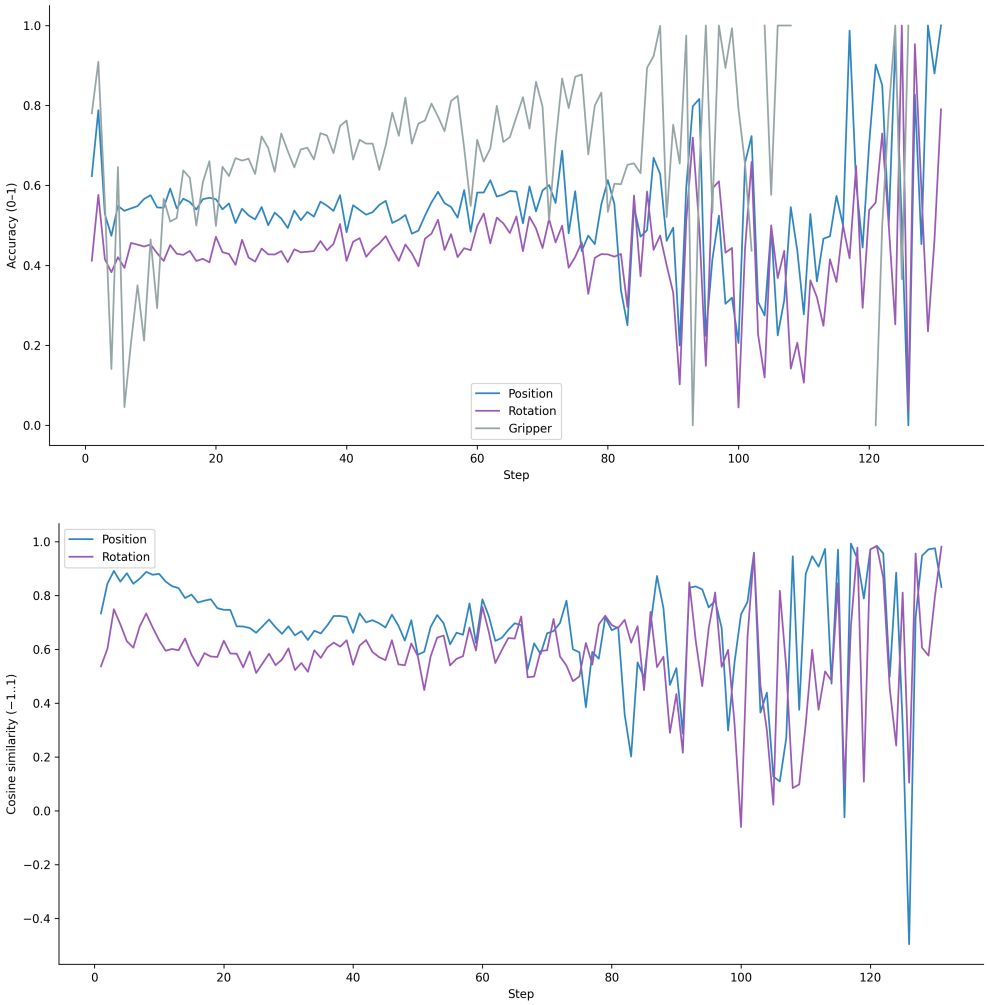


Figure 8.14: Masked-Context Inference: step-wise DirAcc (top) and CosSim (bottom). Position and rotation drop to lower plateaus and become more variable when history tokens are masked, while the gripper retains a similar rising trend to the full-context controller.

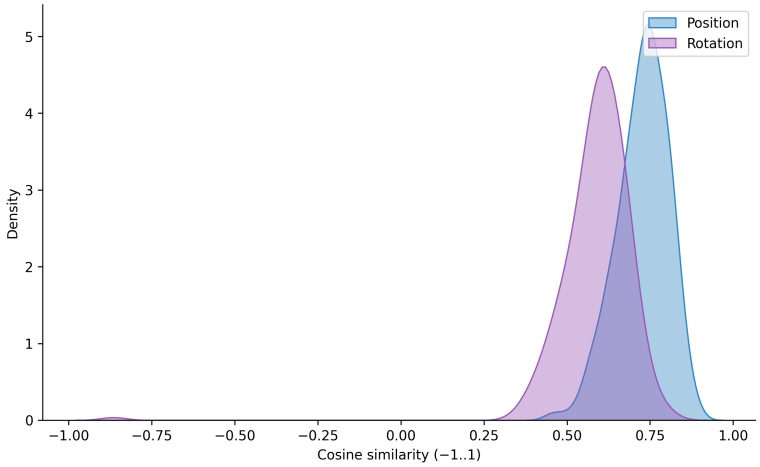


Figure 8.15: Masked-Context Inference: episode-level cosine similarity distributions. Compared with full context, both position and rotation distributions broaden and shift left, indicating reduced and less stable directional alignment in action space.

8.4.3 Cross-Attention Analysis

To complement the quantitative evaluation of both the single-frame and multi-frame policies, we performed an extensive qualitative analysis of the cross-attention patterns generated by the LLaMA 3.2 Vision model during inference. The objective of this study is to understand how the model distributes attention across past and current visual observations, how different parts of the prompt influence cross-modal grounding, and how these internal mechanisms relate to the observed behavior in action space. The analysis is conducted on a curated set of samples drawn from the multi-frame evaluation, with additional comparisons against the single-frame baseline.

For each selected example, we extracted cross-attention maps from a predefined set of decoder layers

$$L = \{3,\ 8,\ 13,\ 18,\ 23,\ 28,\ 33,\ 38\},$$

corresponding to evenly spaced checkpoints across the model depth. Attention tensors are obtained by enabling output_attentions=True during inference and averaging the multi-head attention weights over heads, yielding a matrix

$$\mathrm{attn}_{\mathrm{mean}} \in \mathbb{R}^{T_{\mathrm{text}} \times T_{\mathrm{img}}}.$$

For multi-frame inputs, the image-token axis is segmented into per-frame slices and reduced to the first 1601 visual tokens (CLS + 1600 patch tokens). Text tokens are grouped into three semantic spans identified with simple prompt-parsing utilities: History covers the repeated blocks that pair past observations with their executed actions, Focus contains the short grounding sentence that directs attention to the arm, gripper, and nearby objects, and Question corresponds to the final instruction-conditioned query that asks what the robot should do next. For each span and each layer, we visualize both per-token top-K patch activations (highlighting the most attended regions for each selected text token) and global top-K patch activations obtained by averaging attention across all tokens in the span.

This extraction setup enables a detailed examination of how the model distributes attention across past and current frames, how visual regions are linked to the different textual components of the prompt, and how these behaviors evolve across network depth. In the remainder of this subsection, we summarize the qualitative patterns consistently observed across layers, spans, and trajectories.

History Span: Temporal Integration and Frame-wise Grounding. The History span contains up to three previous observations, each paired with their executed actions. Across all examined samples, this span produces the most

distributed form of cross-attention: nearly all layers assign non-negligible attention to every available frame, indicating that the temporal window is indeed being utilized for contextual grounding. Importantly, even when early timesteps contain black placeholder frames, typical in the initial phase of the sliding window, the model still allocates attention mass to these frames. This behavior is expected, as the history sequence is structurally aligned across examples and the model does not explicitly distinguish blank inputs from valid visual content. However, placeholder frames do not introduce meaningful visual gradients, and attention on them contributes little to the action prediction (cf. Figures 8.16 and 8.17). Consequently, predictions at early timesteps (when the temporal window is still incomplete) exhibit reduced stability, a limitation also reflected by higher continuous and action-space errors for initial steps (cf. Figures 8.12 and 8.13).

Despite this limitation, once all history frames become valid, the cross-attention consistently links the visual patches corresponding to the robot arm, manipulated object, and relevant workspace regions to the previously executed actions. This confirms that the model does not only use the history span as text but genuinely integrates multi-frame visual structure into its hidden representations.
Example (episode_15589_34). Instruction: “pick redbull can from middle drawer and place on counter”. The history span shows attention over all frames (Figures 8.16 and 8.17). Despite a token accuracy of only 0.25, each component of the predicted action lies very close to the ground-truth tokens, yielding an almost exact vector match that is rarely observed in the single-frame baseline.

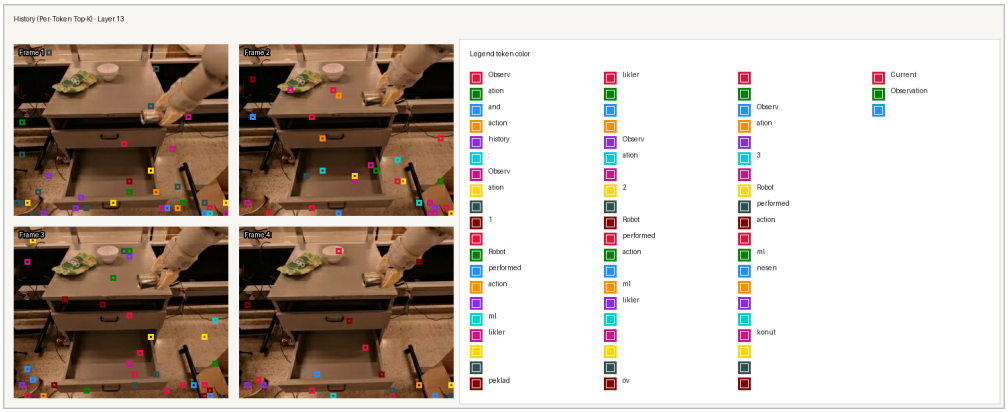


Figure 8.16: History span, Layer 13 (per-token). Attention distributed across all frames with token-wise clusters over arm, drawer and objects. Instruction: “pick redbull can from middle drawer and place on counter”. Token accuracy = 0.25.

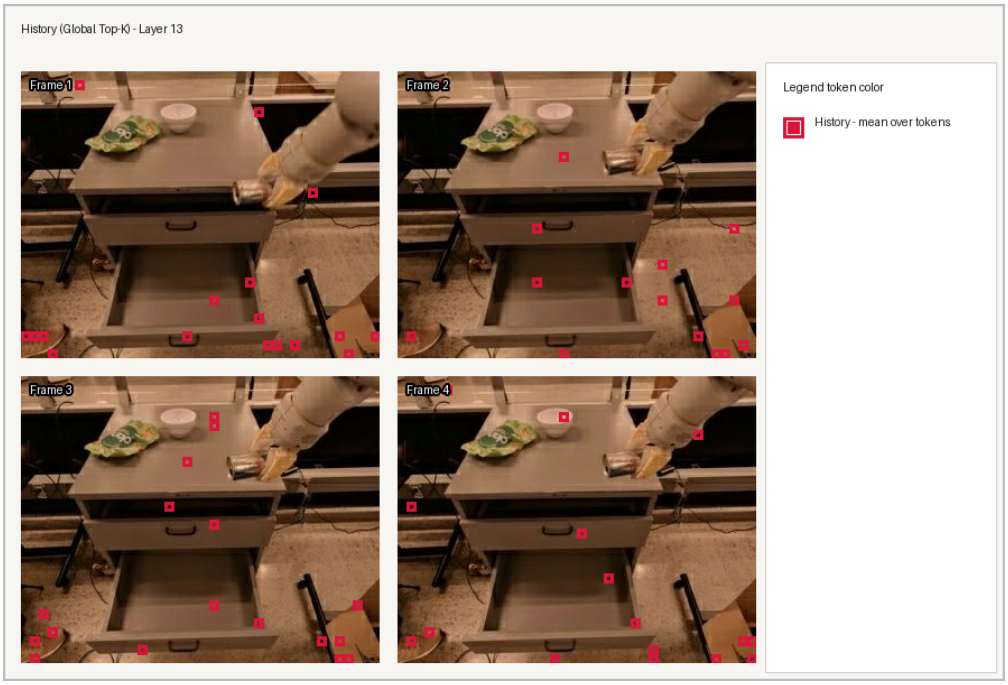
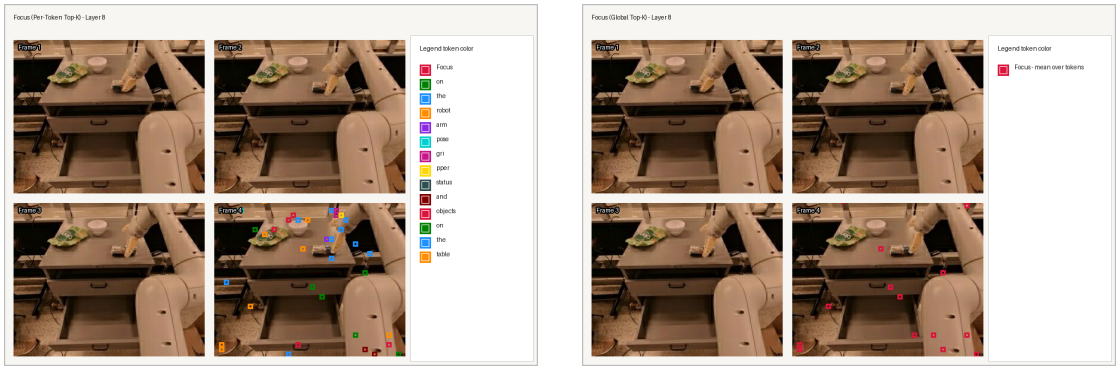


Figure 8.17: History span, Layer 13 (global). All four frames attract non-negligible attention, evidencing temporal integration in the sliding window. Instruction as above.

Focus Span: Spatial Anchoring and Localized Grounding. The Focus span contains a short textual sentence (e.g., "Focus on the robot arm pose, gripper status, and objects on the table") designed to provide a lightweight visual anchor. Across mid-level cross-attention layers, especially layers 8 and 13, the Focus span produces some of the most interpretable patterns observed in the model. Per-token attention maps consistently highlight compact clusters around the robot arm, the gripper, and the immediate neighborhood of the manipulated object. When the scene contains bowls, drawers, or objects partially occluded by the arm, mid-level layers tend to produce stable and semantically consistent localization.

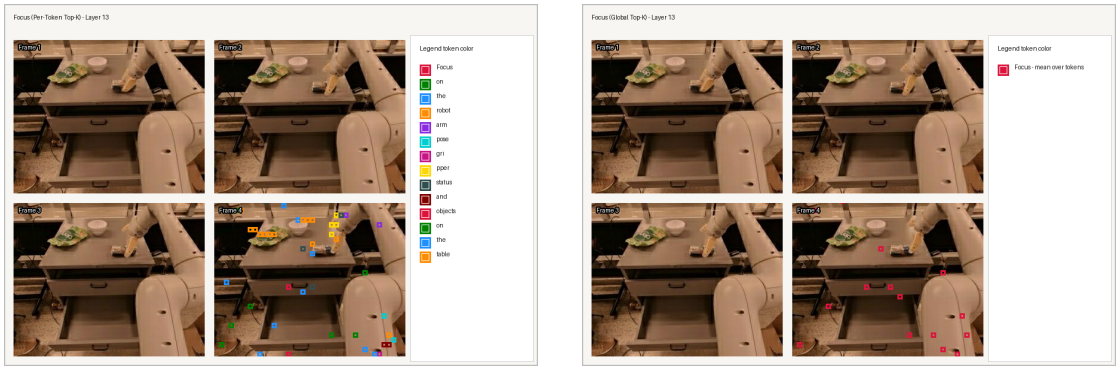
These observations demonstrate that the inclusion of a grounding phrase in the prompt substantially improves visual-textual alignment, an effect already noted in the ALFRED experiments of Chapter 7. See Figures 8.18 and 8.19 for representative mid-level examples. The Focus span therefore acts as a bridge between the temporal visual context provided by the history and the goal-conditioned reasoning captured by the question.
Examples.

- episode_15589_44 ("pick redbull can ..."): perfect alignment, token accuracy 1.00, sequence match true. Focus maps at layers 08/13 show compact clusters on arm, gripper, drawer and can (Figures 8.18 and 8.19).
- episode_14829_37 ("pick apple ..."): gripper parzialmente occluso e localizzazione leggermente offset. Focus L13 per-token evidenzia l'area corretta ma la sequenza è errata: Pred [1, 132, 130, 129, 126, 130, 123, 128] vs GT [1, 132, 133, 127, 122, 131, 123, 0] (token accuracy 0.375); cf. Figure 8.24.



(a) Per-token **(b)** Global

Figure 8.18: Focus span, Layer 08. Per-token and global attention concentrate on arm, gripper, drawer and the can, yielding compact and consistent localization. Instruction: "pick redbull can from middle drawer and place on counter". Token accuracy = 1.00.



(a) Per-token **(b)** Global

Figure 8.19: Focus span, Layer 13. Strong object-centric clusters around gripper and manipulated objects; stable spatial anchoring across frames. Same instruction as above.

Question Span: Goal-Conditioned Integration. The final span, which introduces the current frame and the natural-language instruction, generates some of the most semantically aligned cross-attention patterns. Per-token maps at mid-level layers (in particular layer 13) frequently place coloured boxes for object- and goal-related tokens directly over the emerging grasp region, while other tokens activate more scattered patches that behave as structured noise. This behavior is consistent with the general picture described by Frai et al. [30], where early multimodal layers jointly encode spatial correspondences and emerging semantic structure. See Figures 8.20 and 8.21 for representative examples.
Example (episode_15296_10). Instruction: "pick blue chip bag from top drawer and place on counter". Pred [1, 133, 122, 113, 107, 120, 139, 128] vs GT [1, 132, 122, 114, 107, 121, 142, 128] (token accuracy 0.50). The Question-span per-token map at layer 13 concentrates several object tokens around the blue chip bag and the gripper in the final frame (Figure 8.20).

An analogous pattern appears in episode_14873_35, where the cluster of question tokens progressively moves onto the orange can despite residual scattered activations in the background (Figure 8.21).

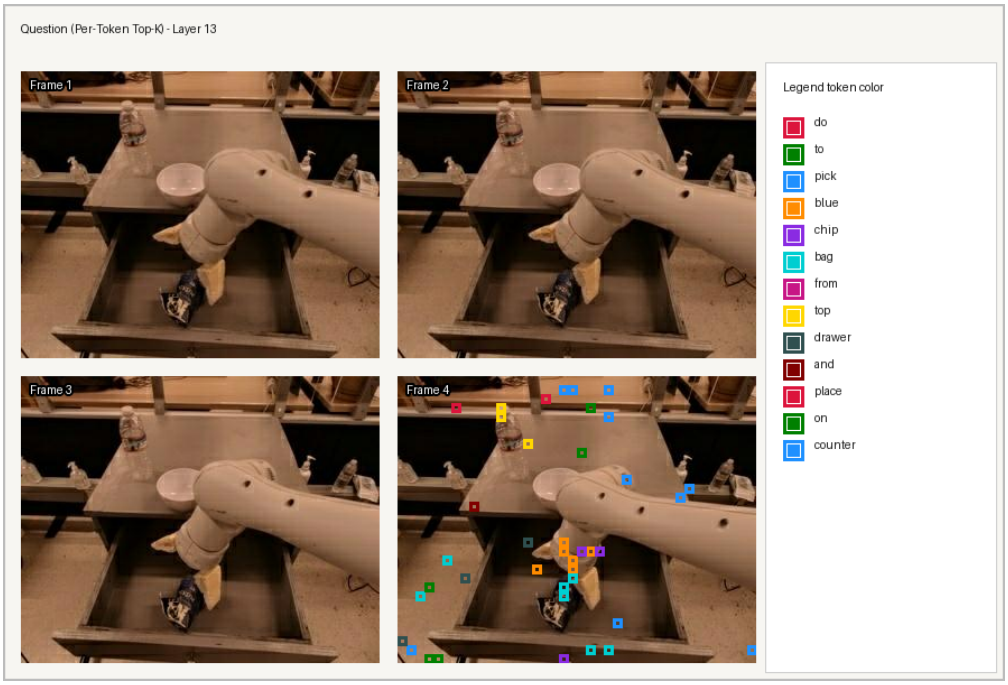


Figure 8.20: Question span, Layer 13 (per-token). Coloured boxes corresponding to object- and goal-related tokens cluster around the blue chip bag and gripper in the final frame, while a subset of tokens activates scattered patches elsewhere in the scene. Instruction: "pick blue chip bag from top drawer and place on counter". Token accuracy = 0.50.

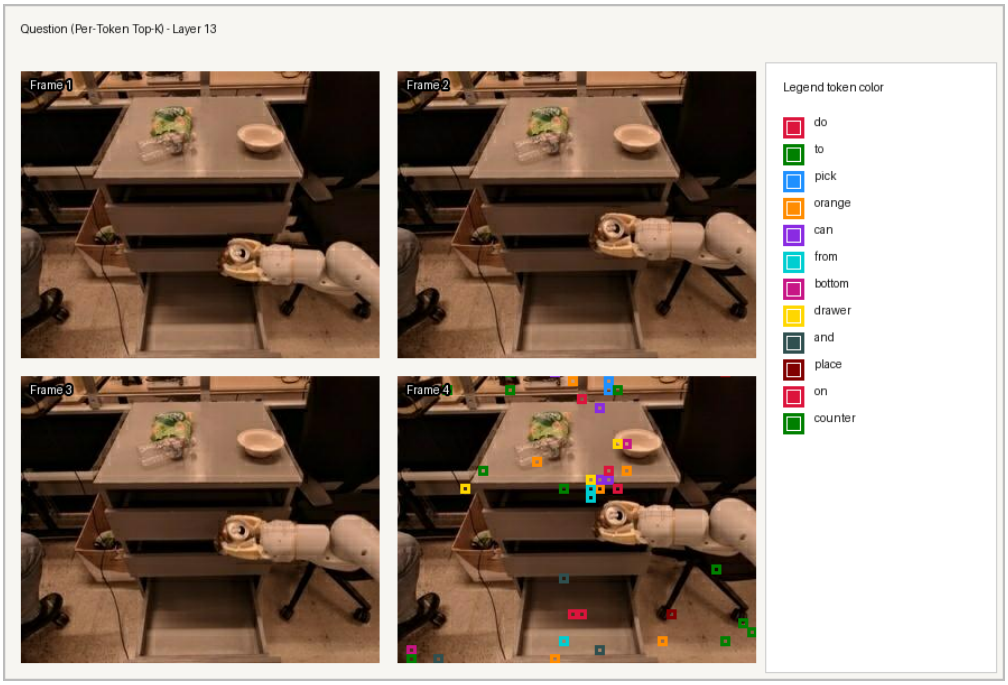
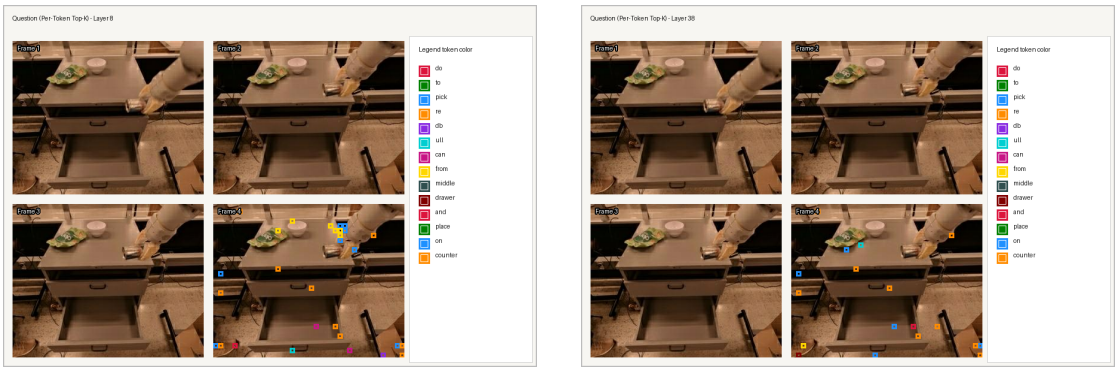


Figure 8.21: Question span, Layer 13 (per-token). Attention shifts toward the grasp region around the orange can despite residual scattered activations that resemble structured noise. Instruction: "pick orange can from bottom drawer and place on counter". Token accuracy = 0.50.

Layer-wise Behavior and Relation to Prior Work. Across depth, cross-attention exhibits a consistent progression. Mid-level layers (particularly 8 and 13) yield the sharpest, object-centric patterns: in the Question span for episode_15589_34, layer 08 concentrates almost all per-token boxes around the gripper and the redbull can in the final frame. Deeper layers (23–38) distribute mass more broadly and semantically, integrating instruction tokens with the global visual context; in the same example, layer 38 spreads activations over a wider portion of the scene while retaining only a weaker focus near the grasp (Figure 8.22). This transition from spatial grounding to abstract multimodal representations aligns with the layered view proposed by Frai et al. [30] and with the action-space improvements observed for the multi-frame policy.



(a) Layer 08 (per-token) **(b)** Layer 38 (per-token)

Figure 8.22: Question span, episode_15589_34. Per-token attention at Layer 08 concentrates around the gripper and target can in the final frame, while Layer 38 becomes more diffuse across the scene but retains some mass near the grasp. Instruction: "pick redbull can from middle drawer and place on counter".

Failure Modes and Attention-driven Misalignment. While the majority of examined samples show consistent alignment between attention patterns and predicted actions, two recurring failure modes emerge. First, when early frames contain black placeholders or visually ambiguous content, cross-attention may incorrectly assign weight to these frames. Although the model continues to perform reasonably well once the window is filled, these early-frame pathologies sometimes cause near-static predictions (e.g., choosing the most frequent token value across components) or incorrect directional deltas; see Figure 8.23. In episode_14873_0 the placeholder frames coincide with poor alignment: Pred [1, 128, 128, 128, 128, 128, 128, 128] vs GT [1, 126, 133, 111, 146, 132, 129, 128] (token accuracy 0.25).

Second, when the gripper or manipulated object is partially occluded or positioned near the edge of the frame, mid-level cross-attention may localize slightly offset from the true object location. In such cases, the resulting action tends to exhibit correct motion direction but incorrect gripper configuration, highlighting a limitation of relying solely on visual grounding without explicit state encoding. These behaviors are consistent with the gripper's higher sensitivity observed in the continuous and discrete metrics of Section 8.4.2. The occlusion/offset case is illustrated in Figure 8.24 (episode_14829_37).

Single-frame Baseline Reference. In absence of temporal context, attention may still hit salient regions but fails to produce coherent sequences; cf. Figures 8.25 and 8.26. In episode_3132_24 ("pick apple from white bowl"), Pred [1, 127, 127, 127, 127, 127, 127, 0] vs GT [1, 128, 128, 128, 128, 128, 128, 220] (token accuracy 0.125).

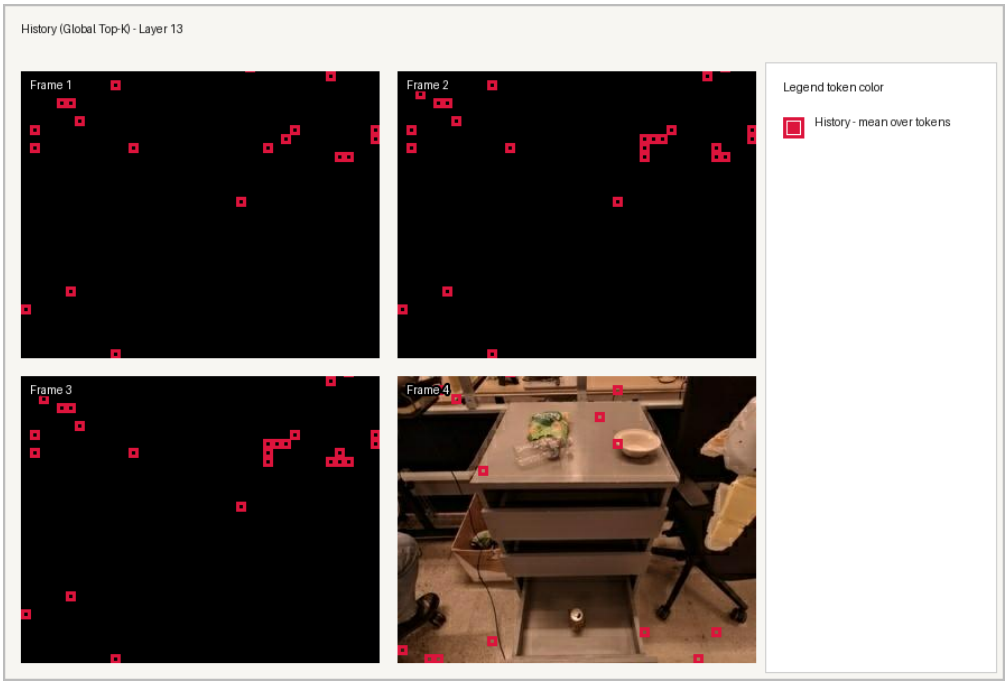


Figure 8.23: Failure mode: Placeholder frames. History span (Layer 13, global) assigns attention to early black frames, which contributes little to action prediction. Token accuracy = 0.25.

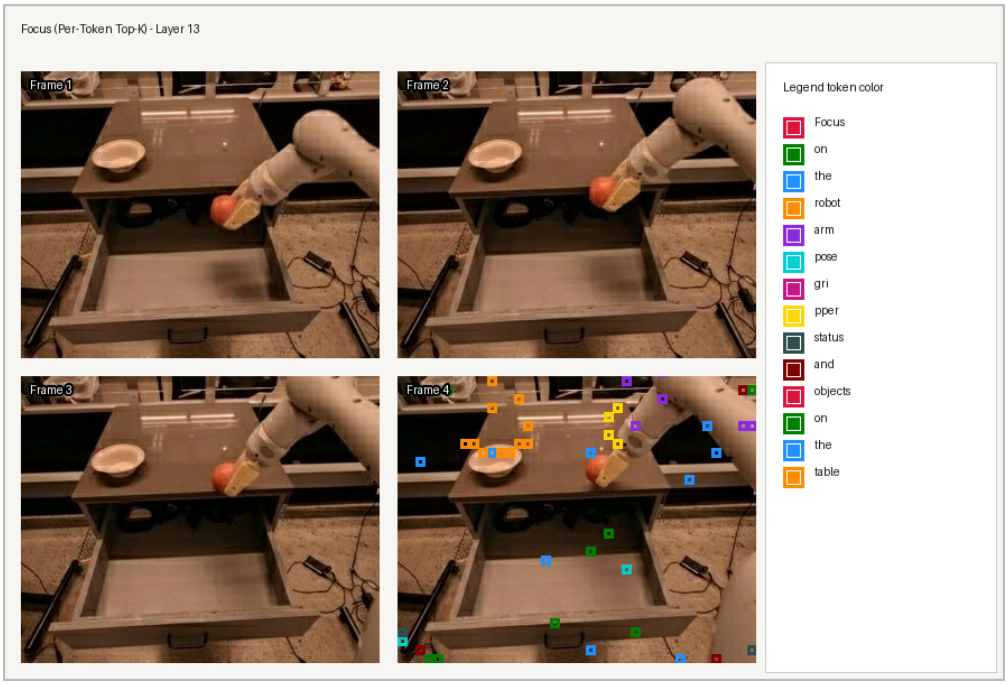


Figure 8.24: Failure mode: Occlusion/offset. Focus span (Layer 13, per-token) localizes near arm and scene objects but gripper is partially occluded and attention is slightly offset from the apple; the predicted sequence is wrong (token accuracy = 0.375).

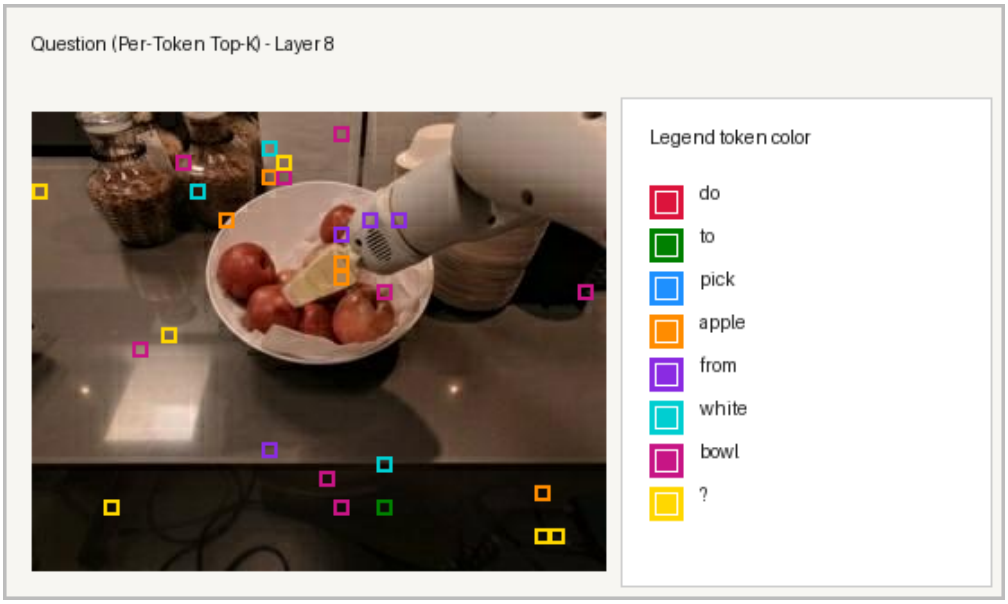


Figure 8.25: Single-frame baseline, Layer 08 (per-token). Attention hits salient regions but lacks temporal grounding; prediction is incoherent (token accuracy = 0.125).

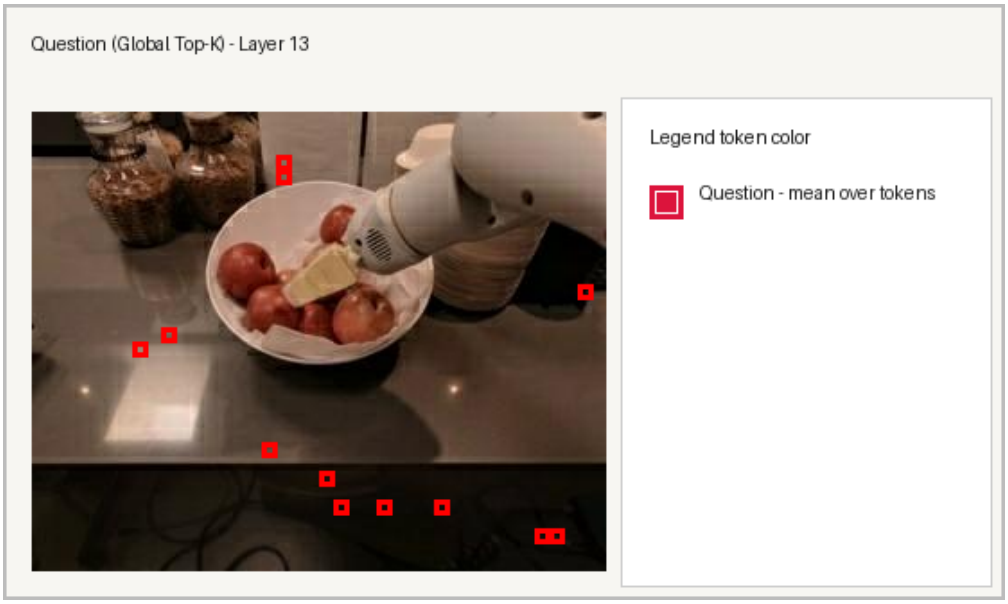


Figure 8.26: Single-frame baseline, Layer 13 (global). Attention appears more diffuse and less spatially grounded compared to multi-frame.

Chapter 9

Quantized Models and 4-bit BnB

9.1 Overview of 4-bit Quantization

Deploying large vision–language models on resource-constrained hardware remains a central challenge in embodied AI. To enable LLaMA 3.2 Vision–Instruct to operate within the memory constraints of a single 80 GiB GPU, we adopt weight-only 4-bit quantization via the BitsAndBytes bnb-int4 backend. Under this scheme, the large linear layers of the model are compressed to 4-bit precision, while activations and all matrix multiplications remain in bfloat16. This hybrid approach preserves numerical stability in the forward pass and maintains compatibility with the standard autoregressive generation interface, while substantially reducing the VRAM footprint of stored model weights.

Concretely, the quantizer employs the NF4 (NormalFloat4) scheme: weights within small blocks are normalized and mapped to one of 16 codebook entries designed to approximate a normal distribution, then rescaled at runtime via learned floating-point scale factors. In our configuration, double quantization of these scales is disabled, leaving scale parameters in floating point. During inference, compressed weights are dequantized on-the-fly into bfloat16 for matrix multiplications and discarded immediately afterward, ensuring that memory overhead remains minimal.

This chapter examines how 4-bit quantization interacts with the two main experimental settings presented in this dissertation: the structured ALFRED planners (Chapter 7) and the continuous Open X controllers (Chapter 8). Rather than exhaustively sweeping quantization configurations, we focus on characterising the practical trade-offs in terms of memory footprint, inference latency, and task-specific performance for a representative NF4 setup widely adopted in contemporary applications.

9.2 Memory and Latency Improvements

Quantizing the LLaMA 3.2 Vision backbone yields measurable gains in both memory footprint and inference time for ALFRED and Open X. Table 9.1 reports the main figures used in this dissertation, comparing full-precision and bnb-int4 variants across domains.

Table 9.1: Summary of memory usage and mean inference time before and after 4-bit bnb-int4 quantization. Memory values refer to peak GPU usage when the model is loaded without inputs (loaded) and during evaluation on the corresponding benchmark (inference).

Domain / model	Precision	Mem. loaded (MiB)	Mem. inference (MiB)	Time (s/sample)
ALFRED Phase B planner	Full	23 183	35 641	129.0
ALFRED Phase B planner	4-bit bnb-int4	11 719	22 689	28.1
Open X single-frame controller	Full	23 183	23 044	0.63
Open X multi-frame controller	Full	23 183	28 636	1.88
Open X multi-frame controller	4-bit bnb-int4	11 719	14 730	1.63

ALFRED Phase B Planner

In the ALFRED experiments, quantization is applied after LoRA fine-tuning of the Phase B structured planner. The full-precision model, when evaluated with 15-frame inputs and JSON output, peaks at roughly 35 641 MiB on the 80 GiB GPU, corresponding to about 43.5% of available memory and a mean inference time of 129.0 s per episode (Section 7.6). The 4-bit bnb-int4 version reduces the loaded footprint to approximately 11 719 MiB (14.3%) and the inference peak to around 22 689 MiB (27.7%), freeing more than 12 GiB of VRAM while preserving the same decoding protocol.

The time–length relationship for the quantized planner is illustrated in Figure 9.1. Most samples cluster between 25 s and 50 s of inference time, with a smaller band of outliers around 120–130 s. The average inference time drops to 28.07 s, a reduction of more than four times compared to the full-precision planner. However, this speed-up comes at the cost of reduced plan quality, a trade-off that is analyzed in more detail in the dedicated quantized-planner section below.

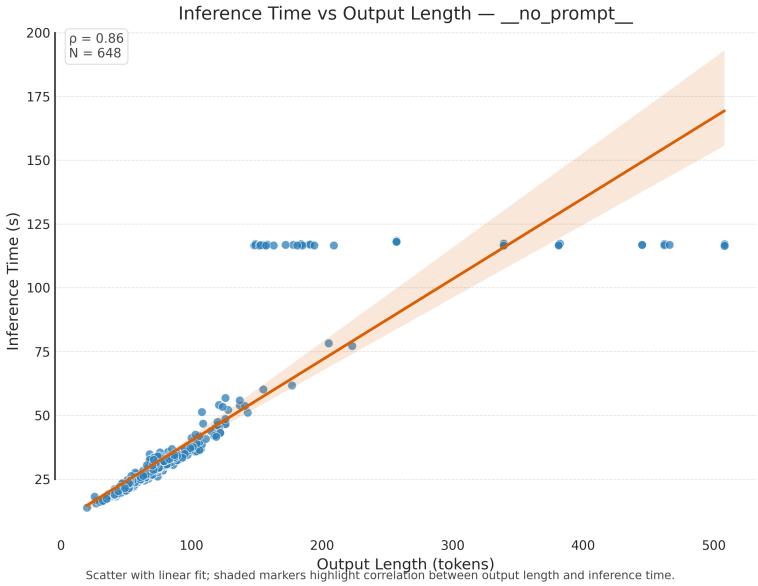


Figure 9.1: Quantised ALFRED Phase B planner (4-bit bnb-int4): inference time versus output length on the TEST-UNSEEN split. The majority of samples lie between 25 s and 50 s, with a smaller group of outliers around 125 s, and a Pearson correlation of 0.86 between sequence length and latency.

Open X-Embodiment Controllers

For Open X, quantization is applied to the multi-frame control policy, which is the most demanding configuration in terms of memory and computation. The unquantized multi-frame controller reaches a peak of about 28 636 MiB during inference, compared with roughly 23 044 MiB for the single-frame baseline. In terms of latency, single-frame decoding averages around 0.63 s per sample (about 1.6 Hz), whereas the multi-frame policy slows down to approximately 1.88 s per sample (about 0.53 Hz), reflecting the cost of processing four images and a longer temporal prompt at each step.

The bnb-int4 multi-frame model substantially reduces the memory footprint, peaking at roughly 14 730 MiB during inference and using only 11 719 MiB when loaded without inputs. Despite this compression, the quantized controller maintains a mean inference time of about 1.63 s per sample (approximately 0.61 Hz), slightly faster than its full-precision counterpart on the same hardware. The behavior of discrete and continuous error metrics for the quantized controller, together with its action-space alignment, is examined in the dedicated Open X analysis section later in this chapter.

9.3 Analysis of the Quantized ALFRED Planner

The ALFRED Phase B planner generates a structured JSON output that first contains a natural language plan and then a discrete action list with symbolic actions and object arguments. Evaluation follows the same lexical, semantic, and discrete action diagnostics introduced in Chapter 7 and is carried out on the ALFRED TEST-UNSEEN split. After quantization, the natural language component remains comparatively robust, while the discrete action list shows a marked degradation, especially in its ability to recover arguments and coherent sequences.

Natural language plan: lexical and semantic analysis. Table 9.2 summarizes lexical metrics for the full-precision and quantized Phase B planners. Quantization slightly reduces BLEU (from 0.29 to 0.24), ROUGE scores (for example ROUGE-L from 0.53 to 0.49), and the composite Average Score (from 0.52 to 0.49). Table 9.3 reports the corresponding semantic metrics: plan-level cosine similarity (STS) remains around 0.85, step-wise similarity stays close to 0.68–0.74, and Coverage@0.7 and STC decrease only marginally. At the same time, the average token count shrinks from about 107 to 68 tokens per plan, and the mean inference time decreases from roughly 129 s to 28 s per episode, in line with the latency gains discussed earlier. Taken together, these trends suggest that the quantized planner continues to behave reasonably well on natural-language plan

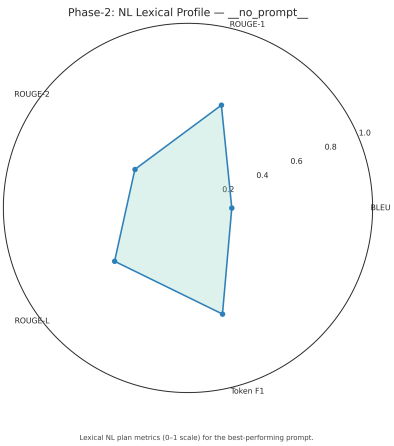
generation, in contrast to the more severe degradation observed for the discrete action branch.

Table 9.2: Lexical metrics for the ALFRED Phase B planner (best prompt).

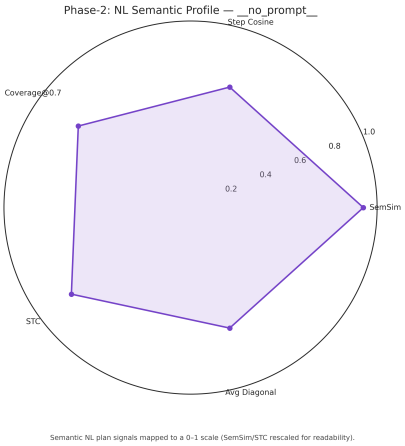
Metric	Full precision	4-bit bnb-int4
BLEU	0.290	0.237
Token F1	0.629	0.604
ROUGE-1	0.618	0.585
ROUGE-2	0.379	0.355
ROUGE-L	0.532	0.492
Average Score	0.522	0.489

Table 9.3: Semantic metrics for the ALFRED Phase B planner on the test-unseen split (best prompt).

Metric	Full precision	4-bit bnb-int4
STS	0.861	0.852
SSS	0.742	0.679
Coverage@0.7	0.806	0.744
STC	0.664	0.577



Lexical metrics



Semantic metrics

Figure 9.2: Lexical and semantic radar plots for the quantized ALFRED Phase B planner on the test-unseen split. The overall shape remains similar to the full-precision model, with moderate shrinkage along lexical axes and near-identical semantic alignment.

Discrete action list analysis. The discrete action metrics in Table 9.4 and the radar in Figure 9.4 exhibit a clear failure mode. Relative to the full-precision planner, action type similarity drops from 0.96 to 0.67 and sequence similarity from 0.88 to 0.54, while action argument similarity falls from 0.87 to 0.00. In the radar plot this corresponds to a global shrinkage of the polygon and, in particular, to the arguments vertex collapsing to zero. The combined action score and the action-specific comprehensive score decrease from about 0.90 to 0.41. Per-episode diagnostics and the example in Figure 9.3 show that the quantized model often predicts a flat list of action labels such as GotoLocation, PickupObject, PutObject, PickupObject, PutObject, instead of a list of JSON objects with explicit arguments like countertop, spoon, fridge, or mug.

Table 9.4: Discrete action metrics for the ALFRED Phase B planner on the test-unseen split (best prompt).

Metric	Full precision	4-bit bnb-int4
Action type similarity	0.955	0.675
Action sequence similarity	0.883	0.543
Action argument similarity	0.865	0.000
Action comprehensive score	0.901	0.406

Instruction

```
Put a mug with a spoon in it on the counter.
```

Ground-truth discrete action list

```
[
  {"action": "GotoLocation", "args": ["countertop"]},
  {"action": "PickupObject", "args": ["spoon"]},
  {"action": "GotoLocation", "args": ["mug"]},
  {"action": "PutObject", "args": ["spoon", "mug"]},
  {"action": "PickupObject", "args": ["mug"]},
  {"action": "GotoLocation", "args": ["countertop"]},
  {"action": "PutObject", "args": ["mug", "countertop"]},
  {"action": "NoOp", "args": []}
]
```

Quantized model output (comprehensive score = 0.39)

```
{
  "nl_plan": "Turn left and go to the counter with the toaster on it. ; Pick up
  the spoon on the counter. ; Put the spoon in the mug on the counter. ; Pick up
  the mug with the spoon in it. ; Put the mug with the spoon in it on the counter.",
  "discrete_action_list": [
    "GotoLocation",
    "PickupObject",
    "PutObject",
    "PickupObject",
    "PutObject"
  ]
}
```

Figure 9.3: Representative sample from the test-unseen split. The quantized planner produces a fluent natural language plan but collapses the discrete action list to a flat sequence of action types without arguments.

The impact on sequence-level structure is visible in Figure 9.5, which reports the distribution of sequence coherence scores between predicted and reference action lists. Compared with the full-precision planner, the quantized model produces many more low-coherence episodes and far fewer high-coherence ones, indicating that action sequences more often drift from the reference order, contain redundant repetitions, and fail to emit the terminating step, the NoOp string. Together with the near-zero argument similarity, this confirms that the discrete branch has undergone substantial forgetting: the model retains only a coarse scaffold of action types while largely failing to recover object bindings and precise step sequences.

One possible interpretation is that weight-only 4-bit quantization disproportionately harms the structured discrete-action head that populates the discrete action field of the JSON output. The planner is trained to emit a single JSON object with one field for the natural-language plan and one for the discrete action list, and the quantized model appears to prioritize the linguistic component while defaulting to underspecified patterns for the symbolic component, such as emitting action names without arguments. This explanation is consistent with the observed metrics and examples but remains a hypothesis; isolating the relative contribution of output structure, loss weighting, and quantization configuration would require targeted experiments.

Future work could address this imbalance by reweighting the loss toward the discrete action segment, training specialized heads or prompts that place the action list earlier in the sequence, or fine-tuning directly on quantized weights with a curriculum that emphasizes argument recovery and sequence coherence. Given that the natural language plan remains stable under NF4 quantization, these interventions could focus on restoring structured control information without sacrificing the efficiency gains reported in Chapter 9.

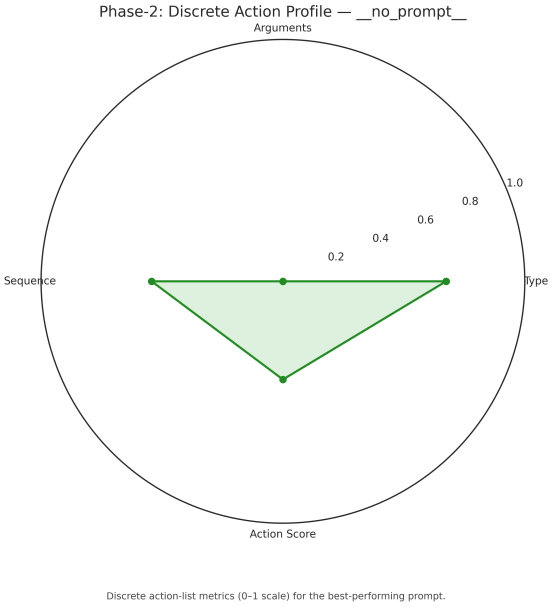


Figure 9.4: Discrete action radar plot for the quantized ALFRED Phase B planner on the test-unseen split. Action type similarity remains moderate, but action arguments and sequence metrics are strongly degraded relative to the full-precision model.

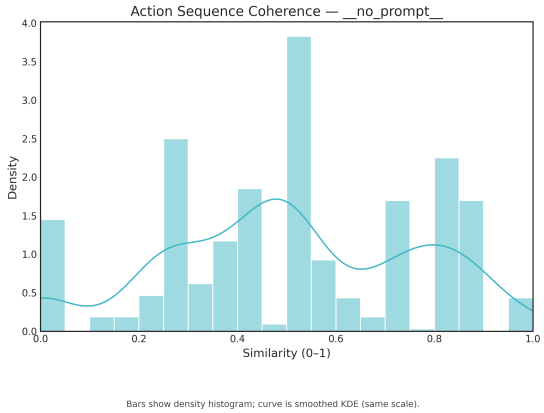


Figure 9.5: Sequence coherence for the discrete action list of the quantized ALFRED Phase B planner. Many episodes exhibit low similarity between predicted and ground-truth sequences, reflecting missing arguments, repeated actions, and altered ordering.

9.4 Analysis of the Quantized OpenX Controller

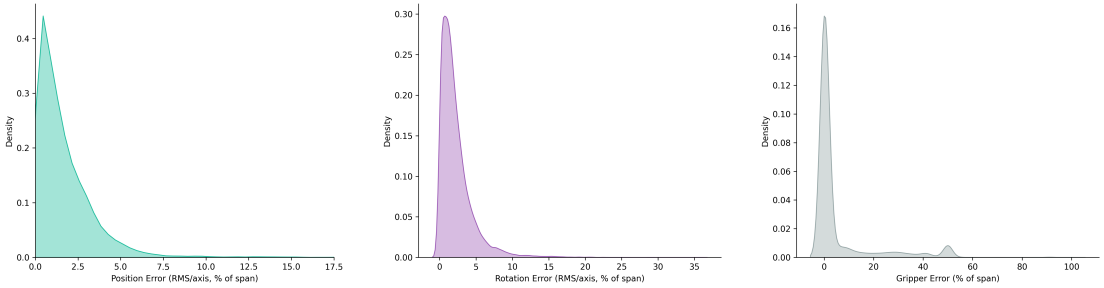
This section analyzes the quantized Open X multi-frame controller, comparing discrete accuracies, continuous errors, action-space metrics, and termination detection performance against the full-precision baseline reported in Chapter 8.

Discrete and continuous metrics. Table 9.5 compares primary action metrics for the full-precision and quantized multi-frame policies. At a first glance, the two models behave very similarly: discrete accuracies for position and rotation remain close (from 24.6% to 22.8% and from 20.0% to 18.1%, respectively), and gripper accuracy stays high, with only a modest reduction (from 77.4% to 75.8%). Continuous errors show the same pattern. The mean positional deviation rises only slightly (from 0.0517 to 0.0581 in normalized units), and the mean rotational deviation from 0.116 to 0.123 rad. In RMS-per-axis form, position and rotation errors increase from 1.49% and 2.13% to 1.68% and 2.26%, while the gripper error grows from 4.99% to 5.50%.

Figure 9.6 confirms this impression visually: the relative error distributions for position, rotation, and gripper are very similar to those of the non-quantized controller reported in Chapter 8, with only a mild broadening of the peaks. On aggregate, the quantized model therefore appears to preserve the continuous accuracy of the full-precision policy. The main discrepancy lies in the exact-match metric, which drops from 3.41% to 0.55%, and, as will be discussed in the following paragraphs, in the temporal behavior of the gripper and in the degraded termination detection, both of which contribute to mismatches at the level of complete 8-token actions.

Table 9.5: Primary quantitative metrics for the Open X-Embodiment multi-frame controller: full-precision versus 4-bit bnb-int4 quantized model.

Metric	Full precision	4-bit bnb-int4
Exact match (%)	3.41	0.55
Token accuracy (mean, %)	38.68	36.86
Position accuracy (discrete, %)	24.62	22.77
Rotation accuracy (discrete, %)	19.97	18.06
Gripper accuracy (discrete, %)	77.39	75.83
Position error (abs, unitless)	0.0517	0.0581
Rotation error (abs, rad)	0.1159	0.1232
Gripper error (abs, unitless)	0.0999	0.1101
Position error (RMS/axis, %)	1.49	1.68
Rotation error (RMS/axis, %)	2.13	2.26
Gripper error (relative, %)	4.99	5.50



Position error (RMS/axis, percent of span).

Rotation error (RMS/axis, percent of span).

Gripper error (relative, percent of span).

Figure 9.6: Relative error distributions for the quantised multi-frame controller. Position and rotation remain close to the full-precision model, while the gripper retains a sharp peak near zero with a slightly heavier tail.

Action-space metrics and gripper dynamics. Action-space alignment metrics for the quantized controller are summarised in Table 9.6. Directional accuracy for position remains very close to the full-precision value (from 0.62 to 0.59), and, interestingly, rotational directional accuracy even increases slightly (from 0.51 to 0.53). Cosine similarities decrease only marginally (position from 0.82 to 0.79, rotation from 0.73 to 0.72). The corresponding mean angular deviations grow by just a few degrees, from roughly 34.6° and 42.7° to 37.0° and 44.0°, indicating that spatial commands remain strongly aligned with the ground-truth action directions after quantization.

In contrast, the gripper branch degrades much more markedly. While the per-step gripper error remains low (Table 9.5), the directional accuracy for the gripper drops from approximately 0.70 to 0.33. This apparent discrepancy arises because gripper error metrics average absolute deviations over all timesteps, including long stretches where the gripper remains stationary and predictions are close to the ground truth. Directional accuracy, instead, is dominated by steps where the ground-truth gripper signal changes and therefore exposes failures in opening and closing dynamics.

Three transitions from episode 14765 make this effect concrete and are summarised in Figure 9.7. Between steps 33 and 34 the ground-truth gripper opens (from approximately -0.28 to -0.04) and the full-precision model tracks this change, whereas the quantised model moves from about $+0.12$ to -0.05, yielding a small absolute error on the final value but a delta with opposite sign. Between steps 34 and 35 the ground truth opens slightly (from about -0.04 to 0.00); the full-precision model again produces a small positive delta, while the quantised model keeps the gripper almost constant, resulting in an almost zero delta. A

similar pattern appears between steps 15 and 16, where the ground truth opens from roughly 0.35 to 0.04: the full-precision prediction moves in the correct direction, whereas the quantised prediction crosses zero and changes in the opposite direction. In all three cases the instantaneous gripper values stay numerically close to the targets, but the step-to-step variations either have the wrong sign or are too small, causing directional accuracy to collapse precisely on these timesteps.

The episode-level distributions in Figure 9.9 and the step-wise aggregates in Figure 9.8 make this pattern explicit: the position and rotation curves remain in the strong alignment regime, whereas the gripper curve is substantially flattened, signalling degraded temporal coherence in opening and closing phases.

Table 9.6: Action-space alignment metrics for the Open X multi-frame controller: full-precision versus 4-bit bnb-int4 quantized model.

Metric	Full precision	4-bit bnb-int4
Directional accuracy (position)	0.62	0.59
Directional accuracy (rotation)	0.51	0.53
Directional accuracy (gripper)	0.70	0.33
Cosine similarity (position)	0.82	0.79
Cosine similarity (rotation)	0.73	0.72
Mean angle (position, degrees)	34.6	37.0
Mean angle (rotation, degrees)	42.7	44.0

Transition	Δg_{GT}	Δg_{Full}	$\Delta g_{\mathrm{Quant.}}$
Episode 14765, steps 33 $\rightarrow$ 34	+0.24 (opens)	+0.24 (aligns)	−0.17 (closes)
Episode 14765, steps 34 $\rightarrow$ 35	+0.05 (opens)	+0.07 (aligns)	0.00 (no change)
Episode 14765, steps 15 $\rightarrow$ 16	−0.31 (opens)	−0.24 (aligns)	+0.20 (closes)

Figure 9.7: Illustrative gripper transitions from episode 14765. In all three cases the full-precision model follows the ground-truth change, whereas the quantized model produces deltas that are too small or have the wrong sign, despite relatively low frame-wise errors.

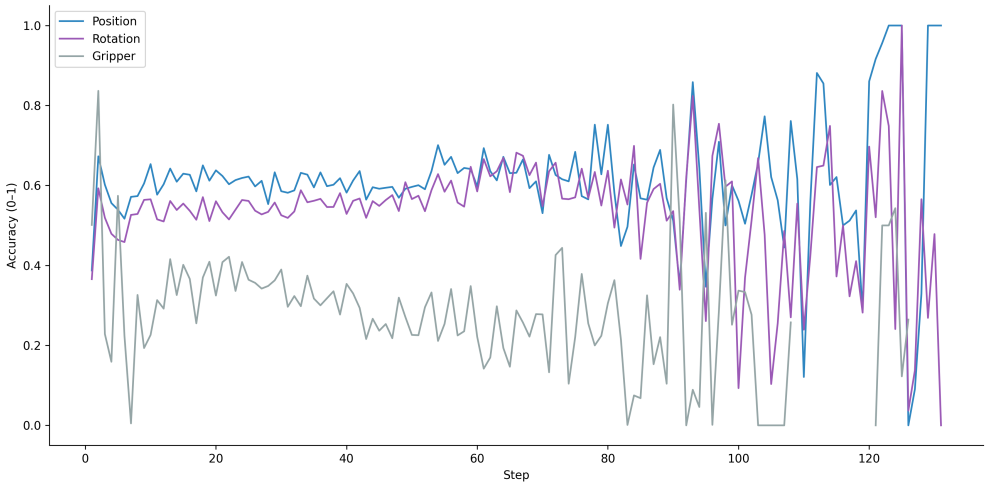
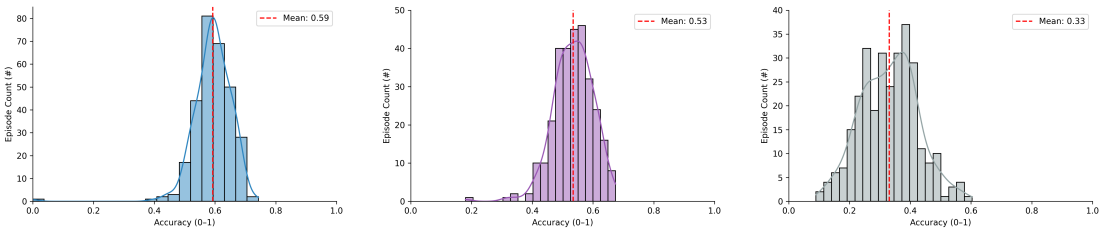


Figure 9.8: Mean directional accuracy over steps for the quantized multi-frame controller. Position and rotation remain close to the full-precision trends, while the gripper curve is noticeably lower than in the non-quantized model, reflecting weaker alignment on opening and closing transitions.



Position Rotation Gripper

Figure 9.9: Episode-level directional accuracy distributions for the quantized multi-frame controller. Position and rotation remain in a strong-alignment regime (means around 0.59 and 0.53), whereas gripper accuracies are shifted toward lower values (mean around 0.33), indicating unstable temporal behavior on gripper transitions.

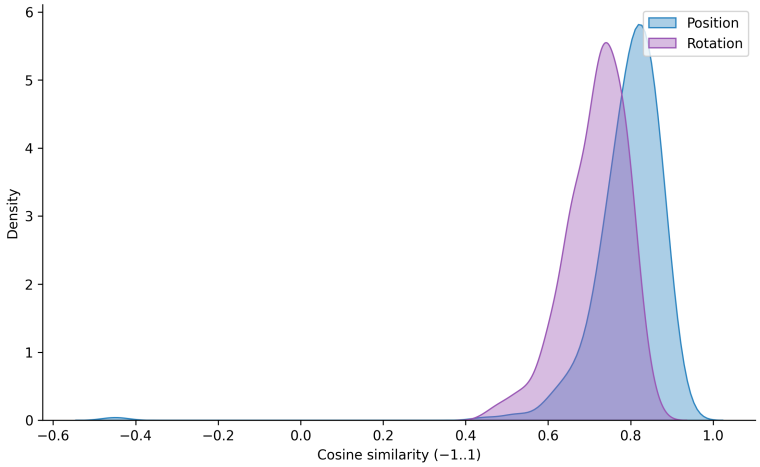


Figure 9.10: Episode-level cosine similarity distributions for the quantized multi-frame controller. Both position and rotation remain strongly aligned with the ground-truth action directions, with densities concentrated between 0.6 and 0.9. The position curve reaches values close to 1.0, while the rotation curve peaks slightly lower, around 0.9, showing only a minor shift relative to the full-precision controller.

Termination detection under quantization. Quantization has a particularly pronounced effect on the termination head. In full precision, the multi-frame controller detects more than half of the termination tokens (precision 99.4%, recall 53.3%, F1 = 69.4%), correctly identifying 320 of the 600 labelled episode endings while producing only two false positives. After 4-bit quantization, precision remains at 100% but recall drops to 5.5% (33 true positives), so most true terminations are missed and episodes are rarely closed by the policy.

This behavior indicates that 4-bit quantization effectively shifts the decision boundary of the termination classifier toward the non-terminating class: logits for the terminate token are suppressed except in a handful of highly confident cases. In practice, the quantized model therefore behaves as if termination were an exceptional event, even in situations where the full-precision controller would reliably emit an end-of-episode flag, suggesting that explicit reweighting or post-hoc calibration of the termination head would be required in a quantized deployment.

Table 9.7: Termination detection metrics for the Open X multi-frame controller: full-precision versus 4-bit bnb-int4 quantized model.

Metric	Full precision	4-bit bnb-int4
Recall (terminate, %)	53.33	5.50
Precision (terminate, %)	99.38	100.0
F1 (terminate, %)	69.41	10.43
Positive support	600	600
Negative support	16,029	16,029

Chapter 10

Conclusions and Future Work

10.1 Conclusions

This dissertation examined the feasibility of using LLaMA 3.2 Vision Instruct as a unified foundation for robotic reasoning and control. Through a series of offline experiments across ALFRED and Open X Embodiment, we explored how a medium sized vision language model can support heterogeneous functionalities ranging from high level task understanding to low level continuous control. The results demonstrate that, when equipped with appropriate prompting structures and lightweight adaptation mechanisms, a single backbone can be specialized to operate meaningfully across multiple levels of abstraction.

A recurring insight throughout the work is the central role of cross attention between visual and textual modalities. In ALFRED, performance improves markedly when training and inference prompts provide a stable, structured context that the model can anchor to: concise preambles with precise instructions, explicit descriptions of the objects in the scene, and a sense of temporal progression introduced through labels such as “Observation 1”, “Observation 2”, and so on. In Open X, where the challenge is even greater, coherent behavior emerges only when the model focuses its attention on key semantic regions of the scene, such as the objects to be manipulated, the gripper, and task relevant spatial cues. Across both domains, the experiments showed that seemingly minor changes in prompt ordering, phrasing, or the grounding between visual and textual inputs can substantially impact performance, underlining how crucial conditioning is for effective VLM based control.

On the ALFRED benchmark, the model was adapted to produce both natural language plans and structured action traces. Under full precision, the system

generates semantically coherent plans and high quality discrete action sequences, suggesting that VLMs can be pushed beyond narrative descriptions to produce representations that approximate executable robotic programs. On Open X, the same backbone functioned as a multi frame controller that begins to capture trajectory structure: continuous errors remain low, and directional alignment improves markedly relative to the single frame baseline.

Despite these strengths, the system remains far from deployable in real world conditions. In ALFRED, the planner emits full programs in a single step, relying on high level primitives such as GotoLocation; a practical robotic agent would require incremental next action prediction, finer grained navigation and manipulation primitives, and explicit mechanisms for state dependent branching. In Open X, the controller exhibits an early understanding of trajectory geometry, but lacks robustness, fails to detect or correct failure modes, and operates at a control frequency unsuitable for real time interaction. These limitations highlight the gap between offline evaluation and embodied deployment.

The scope of the study was also shaped by tight computational constraints, since adaptation relied on LoRA with modest ranks, restricting the number of trainable parameters. Both ALFRED and Open X were subsampled to fit within available resources, and the model was quantized only after fine tuning, without quantization aware training. These choices provided a controlled experimental setting under realistic hardware budgets, but they also imply that current results are a conservative estimate of what such models might achieve under more extensive adaptation.

10.2 Future Work

The limitations identified in this study open several avenues for further research. These can be grouped along two main axes: high-level action planning in symbolic spaces (ALFRED) and low-level continuous control for robot arms (Open X). For each axis, we distinguish near-term extensions that could be implemented within the current framework from longer-term directions that require more substantial changes in modeling or infrastructure.

10.2.1 Action Planning and Symbolic Reasoning

Near-term extensions. On the planning side, several improvements can be pursued without changing the overall architecture. A first step is to move from single-shot program generation to incremental next-action prediction, so that the planner produces one instruction or symbolic action at a time conditioned on the evolving state. This would enable state-dependent corrections, safety checks, and dynamic replanning, rather than committing to a full sequence upfront. Expanding

the discrete action vocabulary with finer-grained primitives (for example short forward motions, small rotations, or incremental gripper commands) would further narrow the gap between high-level GotoLocation-style programs and the control interfaces expected by real robots. Finally, latency and decoding stability can be improved by experimenting with action-only decoding modes, constrained JSON generation, and prompt designs that prioritize symbolic completeness over verbose natural-language plans.

Longer-term directions. In the longer term, action planning should be evaluated in closed-loop settings and on physical or high-fidelity robotic platforms. This entails coupling the planner to navigation and manipulation stacks that execute the generated programs, monitoring execution outcomes, and feeding back failures or deviations as additional supervision. Sim-to-real transfer also remains a key challenge: ALFRED's synthetic environments differ substantially from real households, calling for domain adaptation, visual fine-tuning on robot data, or hybrid datasets that blend simulated and real scenes. Beyond single-task evaluation, multi-task training regimes in which the same backbone plans for multiple embodied benchmarks, with behavior modulated entirely by the prompt or by lightweight adapters, offer a promising route toward general-purpose planners that generalize across tasks, environments, and robot morphologies.

10.2.2 Low-Level Control for Robot Arms

Near-term extensions. For Open X-style continuous control, the most immediate priority is to move beyond purely offline evaluation and study the controller in closed loop, at least in realistic simulators. Running the learned policy so that its actions influence subsequent observations would reveal failure modes that remain hidden in one-step metrics, and would make it possible to design simple recovery strategies, such as re-planning or resetting when trajectories drift too far from the target. At the modeling level, promising short-term improvements include loss reweighting for rare but critical tokens such as the termination flag, better calibration of gripper dynamics, and incorporation of additional state signals as inputs. The RLDS logs expose 7D end-effector poses (position plus quaternion) and orientation descriptors tied to objects or workspace regions; encoding these as continuous features or discretized state tokens would give the model explicit access to the robot configuration while still predicting per-step deltas, potentially yielding more accurate and stable motion. Quantization-aware fine-tuning, structured pruning, or distilled backbones can also be explored to reduce latency and memory footprint without the quality loss observed with naive post-hoc 4-bit quantization.

Longer-term directions. At a broader scale, future work on continuous control should aim at hierarchical, data-efficient policies that bridge language, perception, and low-level actuation. One avenue is to train controllers jointly across multiple embodiments and datasets, using a mixture of shared and robot-specific adapters so that experience on one platform benefits others. Another is to integrate reinforcement learning or closed-loop imitation into the fine-tuning loop, allowing the model to refine its control policy based on task success rather than token-level objectives alone. Scaling up along data, model capacity, and context length, for example by leveraging LLaMA 4 (released in April 2025, with mixture-of-experts routing and longer visual histories), will likely be necessary to capture richer long-horizon behaviors. Any such scaling, however, must remain compatible with efficient deployment: quantization, pruning, and architectural simplification will continue to play a central role in making VLM-based controllers practical for real-world robot arms.

Appendix A

Training Troubleshooting

A.1 Label Smoothing and Loss Computation

During preliminary experiments we explored enabling label smoothing in the Hugging Face Trainer for the MllamaForConditionalGeneration model used in this dissertation. Label smoothing is a regularisation technique that replaces one-hot targets with softened distributions, assigning most of the probability mass to the correct class but reserving a small share for the remaining classes. In causal language modelling this is typically applied after shifting the targets by one position, so that each prediction at time step t is trained against the ground-truth token at time $t + 1$ (next-token prediction).

Under Transformers 4.51.3 we discovered that the automatic label-smoothing logic did not recognise MllamaForConditionalGeneration as a causal language model. Internally, the Trainer consults a registry of causal language-model names, exposed in the library as the constant `MODEL_FOR_CAUSAL_LM_MAPPING_NAMES`, to decide whether labels should be shifted before computing the smoothed loss. Because MllamaForConditionalGeneration was not treated as a causal model in that registry, the Trainer applied label smoothing without shifting the labels.

The relevant pseudocode can be summarised as:

```
if self.label_smoother is not None:
    if model_name in causal_lm_names:
        loss = self.label_smoother(
            outputs, labels, shift_labels=True
        )
    else:
        loss = self.label_smoother(
            outputs, labels
        )  # incorrect for Mllama
```

When label smoothing was enabled from the start of training, the loss appeared to decrease normally and the optimisation curves looked perfectly reasonable. However, generation quality degraded severely: models trained under this configuration produced incoherent token sequences and invalid JSON outputs, despite the apparently convergent loss. The issue became undeniable when resuming from a checkpoint that had been trained with `label_smoothing = 0.0`. That checkpoint exhibited a stable loss around 0.4–0.5 and good qualitative behaviour on validation samples. As soon as training was resumed with `label_smoothing = 0.05`, the reported loss jumped abruptly to values in the 20–30 range and failed to recover, signalling a mismatch between the Trainer's objective and the model's predictions.

To verify the hypothesis we recomputed the loss manually using the standard causal shift:

```
# Correct manual computation (causal shift)
shift_logits = logits[:, :-1, :].contiguous()
shift_labels = labels[:, 1:].contiguous()
loss_fct = nn.CrossEntropyLoss(ignore_index=-100)
manual_loss = loss_fct(
    shift_logits.view(-1, shift_logits.size(-1)),
    shift_labels.view(-1)
)
```

The manually computed loss was consistent with the pre-smoothing checkpoints, confirming that the problem lay in how the Trainer combined label smoothing with causal modelling for this specific architecture, rather than in the model itself.

In principle the bug can be patched by providing a custom loss function that always performs the causal shift before applying smoothing. For the purposes of this dissertation, however, we opted for a simpler and safer solution: all final runs reported in Chapters 7, 8 and 9 were trained with label smoothing disabled (`label_smoothing = 0.0`). This choice avoids subtle training instabilities while keeping the experimental setup compatible with the standard Hugging Face tooling. The discussion in this appendix is meant as a troubleshooting note for future work that may want to revisit label smoothing with updated library versions or custom training loops.

Bibliography

[1] Michael Ahn, Anthony Brohan, Josh Tobin, et al. «Do As I Can, Not As I Say: Grounding Language in Robotic Affordances». In: *Conference on Robot Learning (CoRL)*. 2022 (cit. on pp. 3, 6).

[2] Eric Jang. «Science and Engineering for Learned Robots». In: *https://evjang.com* (2021) (cit. on pp. 3, 4).

[3] Scott Reed et al. «A Generalist Agent». In: *arXiv preprint arXiv:2205.06175* (2022) (cit. on p. 4).

[4] Anthony Brohan et al. «RT-1: Robotics Transformer for Real-World Control at Scale». In: *arXiv preprint arXiv:2212.06817* (2022) (cit. on pp. 4, 5, 19, 21).

[5] Anthony Brohan et al. «RT-2: Vision-Language-Action Models Transfer Web Knowledge to Robotic Control». In: *arXiv preprint arXiv:2307.15818* (2023) (cit. on pp. 4, 5, 19).

[6] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. «BC-Z: Zero-Shot Task Generalization with Robotic Imitation Learning». In: *arXiv preprint arXiv:2202.02005* (2022) (cit. on p. 5).

[7] Danny Driess et al. «PaLM-E: An Embodied Multimodal Language Model». In: *arXiv preprint arXiv:2303.03378* (2023) (cit. on pp. 5, 6).

[8] Danny Driess et al. «Open X-Embodiment: Robotic Learning Datasets and RT-X Models». In: *arXiv preprint arXiv:2310.08864* (2023) (cit. on p. 5).

[9] Jean-Baptiste Alayrac et al. «Flamingo: a Visual Language Model for Few-Shot Learning». In: *arXiv preprint arXiv:2204.14198* (2022) (cit. on p. 7).

[10] Sudeep Dasari et al. «Vision-Language Foundation Models as Effective Robot Imitators». In: *Conference on Robot Learning (CoRL)*. OpenReview: lFYj0oibGR. 2023 (cit. on p. 7).

[11] Yen-Chun Chen et al. «PaLI-X: On Scaling up a Multilingual Vision and Language Model». In: *arXiv preprint arXiv:2305.18565* (2023) (cit. on p. 7).

[12] Jianing Qi. «Inside MLLaMA 3.2: Understanding Meta's Vision-Language Model Architecture». In: *Medium* (Nov. 2024). URL: https://j-qi.medium.com/inside-mllama-3-2-understanding-metas-vision-language-model-architecture-ae12ad24dcbf (cit. on pp. 8, 9).

[13] Mohit Shridhar et al. «ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks». In: *arXiv preprint arXiv:1912.01734* (2020) (cit. on p. 14).

[14] Embodiment Collaboration et al. *Open X-Embodiment: Robotic Learning Datasets and RT-X Models*. 2025. arXiv: 2310.08864 [cs.RO]. URL: https://arxiv.org/abs/2310.08864 (cit. on pp. 18, 19, 21, 29).

[15] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. «BLEU: a method for automatic evaluation of machine translation». In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. ACL. 2002, pp. 311–318 (cit. on p. 43).

[16] Chin-Yew Lin. «ROUGE: A Package for Automatic Evaluation of Summaries». In: *Text summarization branches out*. ACL. 2004, pp. 74–81 (cit. on p. 43).

[17] Nils Reimers and Iryna Gurevych. «Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks». In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. ACL. 2019, pp. 3982–3992 (cit. on pp. 43, 45).

[18] Wenhui Wang, Hangbo Bao, Li Dong, and Furu Wei. «MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers». In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020, pp. 5776–5788 (cit. on p. 43).

[19] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. «SemEval-2017 Task 1: Semantic Textual Similarity Multilingual and Cross-lingual Focused Evaluation». In: *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*. 2017 (cit. on p. 43).

[20] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. «BERTScore: Evaluating Text Generation with BERT». In: *International Conference on Learning Representations*. 2020. URL: https://arxiv.org/abs/1904.09675 (cit. on p. 44).

[21] Tanmay Gupta and Aniruddha Kembhavi. «Visual Programming: Compositional Visual Reasoning without Training». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 14953–14962. DOI: 10.1109/CVPR52729.2023.01436 (cit. on p. 44).

[22] Michael Ahn, Anthony Brohan, Josh Tobin, et al. «Do As I Can, Not As I Say: Grounding Language in Robotic Affordances». In: *Conference on Robot Learning (CoRL)*. 2022 (cit. on p. 44).

[23] Wenlong Huang et al. «Inner Monologue: Embodied Reasoning through Planning with Language Models». In: *arXiv preprint arXiv:2207.05608* (2022) (cit. on p. 44).

[24] Kawin Ethayarajh. «How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings». In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 2019, pp. 55–65 (cit. on p. 45).

[25] Tianyu Gao, Xingcheng Yao, and Danqi Chen. «SimCSE: Simple Contrastive Learning of Sentence Embeddings». In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 6894–6910 (cit. on p. 45).

[26] Bohan Li, Cheng Han, Jianpeng Hou, Yanyan Yang, Yun Ye, and Qiang Yuan. «Sentence Transformations via Whitening for Better Semantic Similarity». In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. 2020, pp. 6055–6067 (cit. on p. 45).

[27] Hao Zhang et al. «Video-LLaMA: An Instruction-tuned Audio-Visual Language Model for Video Understanding». In: *arXiv preprint arXiv:2306.02858* (2023) (cit. on p. 75).

[28] Haotian Liu et al. «LLaVA 1.6: Improved Multi-Modal Instruction Tuning». In: *arXiv preprint arXiv:2404.01258* (2024) (cit. on p. 75).

[29] Wenliang Dai et al. «InstructBLIP: Towards General-Purpose Vision-Language Models with Instruction Tuning». In: *arXiv preprint arXiv:2305.06500* (2023) (cit. on p. 75).

[30] Lior Frai, Eero Simoncelli, et al. «What Vision-Language Models Learn About Semantics». In: *arXiv preprint arXiv:2306.01874* (2023) (cit. on pp. 121, 124).