MATEO: A Multimodal Benchmark for Planning with Temporal Execution Ordering

Anonymous ACL submission

Abstract

To achieve complex real-world goals, AI agents must be able to plan appropriate sequences of actions from a task specification. However, evaluating and improving the planning capabilities is challenging because planning involves multiple steps that are difficult to disentangle when viewed as a whole. A more feasible approach is to focus on Temporal Execution Order (TEO), a core planning sub-task that requires partially ordering actions based on dependency by reasoning about temporal and causal dependency constraints. Existing research studying models' reasoning over temporal execution is limited to automatically derived annotations, approximations of TEO as a linear chain, or text-only inputs. To address this gap, we introduce MATEO (MultimodAl Temporal Execution Order), a multimodal benchmark designed to assess and improve the temporal reasoning abilities of Large Vision Language Models (LVLMs). We acquire a high-quality multimodal recipe corpus, authored through a standardized editorial process that decomposes instructions into discrete steps, each paired with corresponding images. We design a novel set of guidelines and procedures to collect TEO annotations as graphs via crowdsourcing, which can be extended to new recipes, languages, and domains. Using MATEO, we evaluate six state-of-the-art LVLMs across model scales, varying language context, multimodal input structure, and fine-tuning strategies, showing the importance of multimodal inputs as well as limitations of these models.

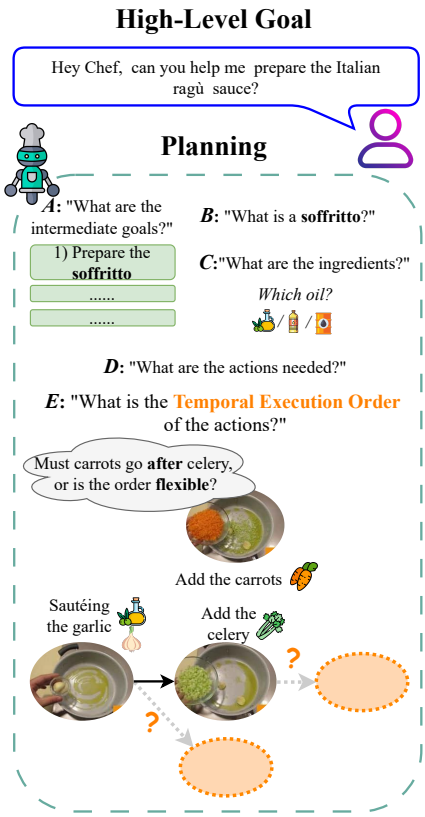


Figure 1: Example of an AI agent's planning process and inherent uncertainties for a natural-language goal. Questions ***A–D*** decompose the high-level goal into executable actions, while ***E*** infers their Temporal Execution Order (TEO) as a directed acyclic graph. Each step introduces uncertainty, producing multiple possible paths, some correct, others wrong.

1 Introduction

The era of autonomous agents with natural language interfaces has recently gained attention from both academia and industry. AI agents have been successfully adopted across many fields, including finance (Fatemi and Hu, 2024; Zhang et al., 2024), software development (Suri et al., 2023; Mo et al., 2025), and networking (Zhang and Zhu, 2023). Agents are autonomous systems that can accomplish goals with little or no human supervision, a concept that dates back to the earliest days of artificial intelligence (Weizenbaum, 1966; Wooldridge and Jennings, 1995). Several frameworks have been proposed to describe the architecture of these systems, including the BDI (belief-desire-intention) model (Rao et al., 1995) or, more recently, the perceive-reason-act-learn paradigm in agentic AI (Raheem and Hossain, 2025). De-

spite differences in terminologies and definitions, these frameworks generally decompose the agent behavior into environmental sensing, reasoning and understanding, planning, and action execution.

Planning in AI agents is one of the most complex components, as it requires understanding high-level goals expressed in natural language, while addressing the inherent ambiguity and uncertainties (Raheem and Hossain, 2025). This process further requires mapping such goals into executable action sequences, as depicted in Figure 1. Constructing such sequences involves selecting appropriate actions and determining the temporal and causal dependencies governing their execution order, i.e., their Temporal Execution Order (TEO). These dependencies can be naturally presented as a Directed Acyclic Graph (DAG), where nodes correspond to actions and edges encode temporal and causal constraints. This structure captures the underlying structure of a plan, where actions are not inherently sequential but partially ordered according to their dependencies.

Given the complexity in understanding high-level goals and reasoning over structured action dependencies in planning, Large Language Models (LLMs) and Large Vision-Language Models (LVLMs) have emerged as promising candidates for real-world planning applications. Nonetheless, current research has shown their serious limitations when dealing with complex tasks such as travel, flight, and calendar planning (Xie et al., 2024; Ji et al., 2025; Zheng et al., 2024). Discovering the root causes of these limitations and identifying possible improvements for the planning task is challenging, as it involves multiple cognitive abilities and is PSPACE-complete (Bylander, 1994).

Conversely, reasoning over TEO (Chambers, 2013; Derczynski, 2017; Vashishtha et al., 2020) is a more tractable problem, as it does not require explicit goal understanding or search over the combinatorial space of possible action sequences. At the same time, it remains a fundamental component of planning, as valid execution critically depends on correctly identifying these dependencies. Specifically, the TEO task involves inferring temporal relations between actions, such as precedence or order independence, by understanding action semantics and reasoning about the temporal and causal constraints governing their execution.

Procedural texts, such as recipes or instruction books, provide an ecologically valid and scalable source of such instances, as they explicitly encode action sequences as plans expressed through stepwise natural-language instructions. However, existing benchmarks rely on text-only procedural text (Mori et al., 2014; Lal et al., 2024), automatically derived annotations, or approximate TEO as a strictly linear chain (Wu et al., 2022; Lu et al., 2024; Qiu et al., 2025). This limits the interpretation of the model's actual performance on the TEO and the insights into whether LVLMs can function as systems capable of grounding plans on world observations (Chen et al., 2025).

In this work, we introduce MATEO (**M**ultimod**A**l **T**emporal **E**xecution **O**rder), a publicly available benchmark for evaluating and improving the multimodal temporal reasoning abilities of LVLMs. MATEO assesses a model's ability to determine the Temporal Execution Order (TEO) of a sequence of multimodal actions, a fundamental requisite for planning in real-world contexts. The benchmark contains 300 high-quality professional recipes from a well-known Italian recipe website[1] listing worldwide recipes. Each recipe is organised as a sequence of steps, with every step containing both a textual description and an image illustrating the action or its outcome, ensuring quasi-perfect semantic alignment between text and visual content. Unlike most existing human-annotated recipe corpora (Mori et al., 2014; Yamakata et al., 2020; Pan et al., 2020), which rely on internal annotators for TEO labelling, we employ crowdsourcing, providing a scalable and reproducible annotation methodology. The annotation process produces a Directed Acyclic Graph (DAG) for each recipe, capturing temporal and causal dependencies (*edges*) of each action (*nodes*). We evaluate six open and closed-sourced LVLMs with various promoting strategies, including in-context learning (Brown et al., 2020), chain of thought (Wei et al., 2022), and self-reflection (Madaan et al., 2023), and fine-tuning approaches. Our findings reveal that while some models achieve state-of-the-art results with multimodal inputs, most still struggle to leverage both modalities. Even the best-performing models achieve 0.76 accuracy and manage to correctly predict the full execution graph only in 3.3%, motivating future adoption of MATEO for developing novel methods to enhance temporal reasoning.

In summary, the main contributions of this paper are:

[1] https://www.giallozafferano.it/

- Release MATEO, a publicly available[2] multimodal benchmark for evaluating and enhancing the temporal reasoning abilities of LVLMs via the Temporal Execution Order (TEO) task;
- A systematic evaluation of six open- and closed-source LVLMs using in-context learning (ICL), chain-of-thought prompting (CoT), self-reflection, and fine-tuning approaches while varying the input modalities;
- A set of novel guidelines for extending the benchmark to new recipes, other languages, and domains, ensuring scalability and reproducibility.

2 Literature Review

Planning The task of planning is to find a sequence of actions that, when executed, transitions a system from an initial state to a desired goal state (Valmeekam et al., 2023). Recent work has investigated LLMs' planning abilities by evaluating them on classical planning problems, such as Blocks World (Gupta et al., 1991), showing their limitations in symbolic planning (Valmeekam et al., 2023; Kambhampati et al., 2024). However, these planning problems might be overspecific and formalized, impeding the model from leveraging any parametric knowledge, and are described with hand-crafted actions and states, impacting the flexibility to adapt to dynamic environments. In this regard, several benchmarks in which initial and goal states are written in natural language have been proposed. They cover various domains such as the planning of trips, meetings, and calendars (Zheng et al., 2024; Xie et al., 2024). Nonetheless, these are only for sequential planning, i.e., without accounting for possible execution of simultaneous actions, and most of the proposed assessments are based on plan-level metrics. A few studies represent the plan as a graph, but they rely on machine-generated annotations (Lin et al., 2024). Although multimodal planning is a relatively new topic, a few benchmarks for investigating the planning abilities of LVLMs have been proposed, such as MPCC (Ji et al., 2025), EgoPlan-Bench2 (Qiu et al., 2025), ALFWorld (Li et al., 2024b), and procedural text corpora Wu et al. (2022); Lu et al. (2024). However, all of these are limited to a linear chain of action due to goals with low uncertainty or a lack of an ad-hoc annotation.

[2]We will add the link for the camera-ready version to comply with anonymization rules.

TEO in Recipes Recipes constitute a challenging multimodal planning domain and a rich procedural-text genre for the TEO task, as their steps often form parallelizable sub-plans with partial ordering constraints. Among the earliest efforts to model this planning complexity are the Japanese Recipe Flow Graph (Mori et al., 2014) and the English Recipe Flow Graph (Yamakata et al., 2020) corpora, which model directly the dependencies and effects of each action expressed by verbs, similarly to a dependency parser. Step-level dependencies have been extracted from the English Recipe Flow Graph to create the CaT-Bench corpus (Lal et al., 2024). Although a multimodal recipe corpus following a similar logic has been proposed (Pan et al., 2020), no public multimodal DAG corpus is currently available. Existing resources rely on non-curated web/video data, leading to inconsistent formats[3], and their annotations are produced by non-crowdworkers, limiting scalability and reproducibility.

3 Methodology

To evaluate the reasoning abilities required for planning in LVLMs, we introduce the Temporal Execution Order (TEO) task. We formulate TEO as a three-way classification problem over the temporal execution relations *before* and *after*, and *independent*. These relations are semantically aligned with the *before*, *after*, and *simultaneous* temporal relations defined in Allen's interval algebra (Allen, 1983). However, unlike Allen's algebraic relations, which are derived from comparisons of interval start and end points, our formulation infers temporal execution order mainly from causal dependencies. In particular, if action B requires a condition that is produced by the post-condition (aka effect) of action A, then A must be executed *before* B (equivalently, B must occur *after* A). Conversely, if A and B do not depend on each other's post-conditions, including indirect or transitive dependencies (e.g., $A \rightarrow C \rightarrow B$, where B requires C and C requires A), then the two actions are considered independent, meaning they can be executed in any order. Given a set of action instances $\mathcal{A}$ and their dependency relations $\mathcal{E}$, TEO can be represented as the graph $G = (\mathcal{A}, \mathcal{E})$, where $\mathcal{A}$ are the nodes, and $\mathcal{E}$ are the edges. Specifically, this graph is a directed acyclic graph (DAG) because the dependencies have an orientation and any cycle

[3]e.g., variable image counts per step, mixed unit systems.

would generate a paradox, making the plan infeasible (e.g., an action dependent on its own output).

Similar to Roccabruna et al. (2024) and Lal et al. (2024), for each class, we pose a corresponding ternary (Yes/No/I don't know) question, as presented in Table 1.

TEO class	Question
Before	Must Step A be executed **before** Step B?
After	Must Step A be executed **after** Step B?
Independent	Can Step A and Step B be executed in **parallel**?

Table 1: Temporal Execution Order (TEO) classes and their corresponding ternary questions.

To investigate whether the models can answer these questions, we formalise the following input sequence. Since our corpus consists of recipes, we refer to actions as steps, following the natural structure of a recipe. In our corpus, a step is composed of an image I and a textual part T, formally $S_j = I_j \oplus T_j$. Thus, given two steps, A and B, the model is tasked to answer the three TEO questions of Table 1, denoted as Q_i, using the following input sequence:

$$C \oplus S_A \oplus S_B \oplus Q_1 \oplus Q_2 \oplus Q_3$$

where C is the grounding context, $\oplus$ denotes concatenation with a newline ("\n"). Marker tokens ("Step A picture:", "Step A description:", etc.) precede each image and text component to guide the model further.

The TEO class is then retrieved by parsing the model's output by assigning class $i \in \{before, after, independent\}$, if the model answers *"Yes"* to the corresponding Q_i TEO question and *"No"* to both other Q_j for $j \neq i$. All the other combinations, such as answering *"Yes"* to multiple questions or responding *"I don't know"*, are assigned to the class *Other*. This captures inconsistent behavior, such as simultaneously affirming the *before* and *after* questions, and uncertainty.

In chain of thought experiments (Wei et al., 2022), the model is prompted to directly predict one of the three TEO classes to simplify and shorten the demonstrations provided in the prompt.

4 The Benchmark

Our requirements for building MATEO were to identify a form of procedural text in which the textual and visual components are maximally semantically aligned, enabling reliable multimodal reasoning over execution order. We have found that these criteria are met by the professionally edited recipes on GialloZafferano, a well-known Italian-language website featuring cuisines from around the world. These recipes offer several advantages: i) they follow an editorially-curated narrative and structural format; ii) the images are professionally produced, visually consistent, and captured from a near-uniform top-down perspective, minimizing viewpoint variability; and iii) each recipe is explicitly segmented into an ordered sequence of steps. Each step typically describes a single executable action, and the accompanying image depicts either the action's outcome or the action in progress. Together, these properties reduce multimodal noise sources and enable cleaner corpus utilization [4].

We have annotated the corpus using the Prolific[5] crowdsourcing platform. To balance crowdworker cognitive load, we divided the annotation process into batches of five recipes, with an average completion time of approximately 50 minutes per batch. We considered only native Italian speakers, given that the recipes are in Italian. We set the compensation to £9.60 per hour, which falls within the platform's recommended range.

The annotators were provided with a list of steps in the order of the original recipe. They were tasked with linking these recipe steps into a DAG by following the TEO relations by an ad-hoc UI (Figure 11 and Figure 10 in the Appendix, respectively). To explain the task concisely and effectively, we provided guidelines, a short demo video, and multimodal instructions, along with explanatory examples. The annotation platform and the guidelines[6] are shown in Appendix A.3.

To ensure data quality, annotators were required to pass a qualification test before starting the actual task by achieving at least 65% accuracy on two test recipes. The accuracy is computed as if it were a multi-label classification task, where the examples

[4] Because GialloZafferano's recipes are copyright-protected and not publicly licensed for AI benchmarking, the publisher agreed to support our research initiative by providing a subset of approximately 300 recipes for release to the research community. The recipes were randomly selected via stratified sampling over dish category (e.g., appetizers, main courses, desserts) to ensure balanced coverage across recipe types.

[5] https://www.prolific.com/

[6] They will also be shared on GitHub along with the code used to conduct our experiments.

are the nodes and the labels are the outgoing nodes[7]. The threshold corresponds to the 75th percentile of the score distribution computed from testing 25 crowdworkers not involved in the actual annotation.

In total, 54 annotators[8] annotated 315 recipes, collected in 63 batches. We computed the annotation agreement using MASI distance metrics (Passonneau, 2006), which are specifically designed for measuring agreement on multi-labelling classification tasks. The annotators reached an agreement score of 0.85 (substantial agreement)[9]. Moreover, we manually checked the annotations for specific error patterns, such as graphs with isolated steps or graphs that were fully linear. At the end of this process, we discarded 15 recipes due to poor annotation quality. Hence, the resulting corpus contains 300 recipes.

Preliminary experiments show that some LVLMs perform worse on this task in Italian than in English, probably because these models are trained primarily on English (Geigle et al., 2025). For this, we translated the recipe steps from Italian to English using Llama-3-8B (Grattafiori et al., 2024), given its relatively small model size and acceptable performance. Before translating the full corpus, we manually verified the translations on the test set and found them to be of high quality. Thus, we conducted all the experiments on the English translations. The dataset is split into training, validation, and test sets using stratified sampling by number of steps, with a 70%/10%/20% split, as reported in Table 2.

	Train	Valid	Test
# Recipes	210	30	60
# Steps	3273	477	936
AVG Steps	15.6 ± 3.9	15.9 ± 3.7	15.6 ± 3.8
Branching Factor	1.12 ± 0.43	1.16 ± 0.42	1.11 ± 0.38

Table 2: Splits of the annotated corpus used for training and evaluating LVLMs' capabilities. The table reports the number of samples, the average number of steps, and the branching factor.

5 Experimental settings

5.1 Models

We select six LVLMs that vary in model size, with selection primarily based on their support for sequences of images in the input. Our evaluation includes both open-source, namely, Qwen3-VL-8B, Qwen3-VL-32B (Bai et al., 2025), LLaVA-OneVision-7B (Li et al., 2024a), InternVL3.5-8B, InternVL3.5-38B (Wang et al., 2025), and the closed-source model GPT-5.1[10].

5.2 Dataset

Given the annotated DAG for each recipe, we infer the TEO classes directly from the graph structure. Although a fully exhaustive temporal-relation evaluation could consider all possible pairs of steps, i.e., $\frac{n(n-1)}{2}$ candidate pairs (437,580 in our test set), we restrict the evaluation to relations derived from annotated dependencies. Specifically, step A is labeled as *before* step B if there is a directed edge from A to B. The class *after* is defined as the inverse of *before*; therefore, we generate the corresponding ground-truth examples by swapping steps A and B. Steps A and B are considered *independent* if there is no directed path between them in either direction. The resulting dataset contains 1,573 *independent* relations, 933 *before* relations, and 933 *after* relations.

To evaluate whether models truly understand the relation among steps or instead rely on pre-training biases, we use two versions of the dataset. The first preserves the original annotation order, yielding *before* and *independent* labels. The second reverses each step pair, mapping *before* to *after*, while *independent* labels remain unchanged.

5.3 Prompt settings

We evaluate the planning abilities of LVLMs on the TEO task under different modalities and grounding contexts. Specifically, we compare text-only, image-only, and multimodal inputs, as described in Section 3. Below, we describe the grounding contexts we used to evaluate the models. The example for each prompt template can be found in Table 7 in the Appendix A.3.

Baseline The model is only conditioned on two recipe steps and the three TEO questions.

[7] To give a better visualization, the data structure is $n_1 : [n_2, n_3], .., n_2 : [..]$, where n_2 and n_3 are the outgoing nodes and the "labels" of n_1.

[8] Some of the annotators participated multiple times because we experienced a shortage of annotators willing to perform our task.

[9] The agreement was computed on the examples used for qualifying the annotators. While this may give an overestimation of the actual agreement, it was the best trade-off as we could not afford to have an overlap on the annotation due to the high amount of time needed to complete a batch.

[10] https://platform.openai.com/docs/models/gpt-5.1

Instructions We provide the model with a set of rules that explain the meaning of the TEO classes and instruct the model to respond to each question. The rules are general for the TEO task and are not specific to the cooking domain.

In-Context Learning (ICL) The model is conditioned on the instructions described previously and three in-context examples for each TEO class. Each example consists of a pair of steps, the TEO questions, and the correct answers, accompanied by brief explanations to encourage the model to emulate the latent reasoning process. To limit context length and computation time, all in-context examples are text-only. We prepend a modality-specific instruction noting that examples contain only text descriptions, while the input may include images.

Chain-of-Thought (CoT) Correctly predicting the TEO class requires understanding whether the outcome of one step is a prerequisite for completing the other. In this experiment, we explicitly ask the model to identify pre- and post-conditions of each step and then reason about their dependency. After this reasoning process, the model is tasked to directly predict the TEO class.

Self-Reflection At the end of the CoT, the model is instructed to reflect on its answer by reviewing any inconsistencies or errors in the reasoning process, and then give the final answer.

5.4 Fine-Tuning

Based on the results of the prompting experiments, we have selected the best-performing small-scale model, InternVL3.5-8B, for fine-tuning. We fine-tune the model with LoRA (Hu et al., 2021) on all the modalities following the *Instructions* prompt task. To prevent label imbalance, we swap only dependent step pairs to generate examples for the *after* class. For independent step pairs, swapping does not change the label and would therefore introduce duplicate examples, potentially biasing the model toward the *independent* class. This resulted in 3,499 examples for each of the *before* and *after* classes, and 4,969 for *independent*. Examples of model inputs are provided in Table 8 in the Appendix A.1 along with all the hyperparameters.

6 Results and Error Analysis

To evaluate models' performance in predicting TEO, we consider both the original and the reverse order of each step pair. This setting provides insight into potential pre-training biases and also reflects realistic scenarios in which the true execution order may be unknown. The accuracy metric is based on a consistency criterion: a prediction is considered correct only if the model remains consistent across both the original and swapped step orders. Specifically, for dependent pairs, the model must predict *before* in the original order and *after* when swapped. For independent pairs, the model must predict *independent* in both configurations.

Table 3 reports accuracy scores across prompting strategies, and after fine-tuning. All results are obtained from a single evaluation using greedy decoding to ensure reproducibility. Starting with the prompting strategies, we observe that in the baseline setting, all models perform poorly, with near-zero accuracy, indicating limited zero-shot ability on the TEO task. Adding task instructions, in-context learning, or chain-of-thought improves performance for most models, but the effectiveness of each setting depends on the model; thus, there is no universal best strategy. Furthermore, results show that self-reflection sometimes leads to lower performance compared to CoT, suggesting a limitation of this prompting strategy for some models.

Regarding individual model performance, Qwen3-VL-32B achieves the highest overall accuracy among the open-source LVLMs. When using ICL with image and text inputs, it attains an accuracy of 0.75. However, performance drops sharply to 0.50 under CoT prompting. As the error analysis in Appendix A.2 shows, this drop is largely driven by the model's weak performance on the *after* class. Among models with fewer than 10B parameters, LLaVA-OneVision-7B substantially underperforms, with accuracy at or near zero in most settings. In contrast, InternVL3.5-8B and Qwen3-VL-8B achieve performance comparable to larger-scale models, reaching 0.58 (instructions, image-only) and 0.53 (CoT, image+text), respectively. Although GPT-5.1 achieves state-of-the-art accuracy of 0.76 under self-reflection, other models do not show gains under this strategy. We find that this is largely due to failure to self-reflect or produce a final answer, with failure rates ranging from 9.5% for Qwen3-VL-32B to 81.6% for LLaVA-OneVision-7B. Moreover, when self-reflection produced the correct answer, models largely preserve their original predictions (98.0% for GPT-5.1 and 99.9% for Qwen3-VL-32B).

Although multimodal input leads Qwen models and GPT-5.1 to achieve the best-performing results across all settings, not all models show the same

Model	Modality	Baseline	Instructions	ICL	CoT	Self-Reflect.
GPT-5.1	Image-only	0.12	0.55	0.63	0.72	0.72
	Text-only	0.08	0.3	0.33	0.64	0.62
	Image+Text	**0.27**	**0.61**	0.73	**0.75**	**0.76**
Qwen3-VL-32B	Image-only	0.15	0.26	0.66	0.5	0.49
	Text-only	0.09	0.22	0.62	0.44	0.42
	Image+Text	0.18	0.48	**0.75**	0.5	0.45
InternVL3.5-38B	Image-only	0.01	0.51	0.53	0.49	0.44
	Text-only	0.10	0.50	0.59	0.37	0.39
	Image+Text	0.09	0.51	0.49	0.50	0.47
InternVL3.5-8B	Image-only	0.15	0.58	0.39	0.14	0.19
	Text-only	0.10	0.31	0.46	0.31	0.32
	Image+Text	0.10	0.34	0.32	0.33	0.24
Qwen3-VL-8B	Image-only	0.03	0.14	0.32	0.48	0.48
	Text-only	0.03	0.26	0.31	0.39	0.41
	Image+Text	0.05	0.48	0.44	0.53	0.5
LLaVA-OneVision-7B	Image-only	0.00	0.00	0.00	0.02	0.00
	Text-only	0.00	0.00	0.07	0.22	0.09
	Image+Text	0.00	0.03	0.04	0.09	0.00
InternVL3.5-8B$_{\text{Fine-tuned}}$	Image-only	-	0.68	-	-	-
	Text-only	-	0.61	-	-	-
	Image+Text	-	**0.69**	-	-	-

Table 3: Performance of LVLMs on the TEO task under different prompt strategies and fine-tuning results. We report accuracy by considering the consistency of predictions across the original and swapped orders of steps. The results in bold indicate the best results per strategy; in the second column, the best results for prompting and fine-tuning are bolded separately.

ability to leverage this modality. Indeed, other models struggle to incorporate multimodal information, and we observe a drop in performance compared to single modalities.

After fine-tuning, InternVL3.5-8B achieves an accuracy of 0.69, approaching the performance of GPT-5.1 and Qwen3-VL-32B. Notably, image-only fine-tuning yields better performance than text-only fine-tuning. Possible explanations are that the model cannot fully leverage textual input alone, or that images more precisely capture the underlying actions, whereas textual descriptions may contain superfluous information, such as references to other steps, or lack sufficient contextual information.

6.1 Graph-level analysis

However, pairwise assessment does not return the model's performance at the TEO graph level. To fill this gap, we evaluate their correctness in reconstructing the full TEO graph. Starting from a set of nodes representing the recipe steps, we add directed edges between nodes whenever the model predicts either *Before* or *After* class for a pair of steps. To reflect a realistic scenario, we consider all evaluated pairs, including the evaluation between independent pairs. This setting is challenging because incorrect predictions on these pairs introduce spurious dependencies. Because independent pairs have no corresponding edge in the ground-truth graph, predicting *before* or *after* for such pairs yields a false positive edge, causing the reconstructed graph to contain more edges than the ground truth. Thus, we also perform a simplified analysis in which we restrict the reconstruction to predictions for edges present in the ground-truth graph, i.e., ignoring independent pairs. Moreover, given that even the best-performing models achieve only modest performance at the edge level, we restrict this analysis to predictions made on the original order of steps to eliminate errors due to inconsistencies across the original and swapped orders.

To evaluate the correctness of model predictions at the graph level, we compare the reconstructed graphs with the ground-truth human-annotated graphs. We use Graph Edit Distance (GED), a standard measure for comparing two graphs. Given

Model	Recon. setting	GED (↓)	GSED (↑)	Exact Matches (# (%))
GPT-5.1	All Pairs	6.37 ± 4.45	0.61 ± 0.24	2 (3.3%)
	GT Edges Only	3.15 ± 2.86	0.80 ± 0.16	7 (11.7%)
Qwen3-VL-32B	All Pairs	7.47 ± 5.21	0.54 ± 0.27	1 (1.7%)
	GT Edges Only	3.47 ± 2.83	0.78 ± 0.16	6 (10.0%)
InternVL3.5-8B$_{\text{Fine-tuned}}$	All Pairs	8.93 ± 6.43	0.49 ± 0.26	1 (1.7%)
	GT Edges Only	5.28 ± 3.70	0.68 ± 0.20	5 (8.3%)

Table 4: Recipe-level evaluation comparing reconstructed and ground-truth graphs. The table reports Graph Edit Distance (the lower, the better), graph similarity (the higher, the better), and the number of exact graph matches (i.e., the number of graphs for which all the relations have been predicted correctly). Results are shown for two reconstruction settings: using all predicted dependencies and restricting to ground-truth edges only. To aid interpretation of the reported metrics, Figures 2 and 3 in the Appendix A.3 show examples of reconstructed graphs.

a source graph and a target graph, GED computes the minimum number of edit operations required to transform the source into the target. Moreover, since GED is a distance measure, we normalize it to yield a similarity score in the range [0, 1]. Specifically, we adapt the Tree Similarity of Edit Distance (Song et al., 2024) to the graph setting, which we refer to as Graph Similarity of Edit Distance (GSED):

$$\text{GSED}(G_1, G_2) = \max\left\{1 - \frac{\text{GED}(G_1, G_2)}{\text{MaxNodes}(G_1, G_2)}, 0\right\}$$

where G_1 is the reconstructed graph, G_2 is the ground truth graph, GED is the Graph Edit Distance between G_1 and G_2, $MaxNodes$ is the maximum number of nodes between the two graphs.

The results are presented in Table 4. For the sake of clarity, we report results of the three best-performing models; results for all models are provided in Table 6 in Appendix A.3. For each model, the reported values correspond to its best-performing setting, as identified in Table 3. Overall, we observe that models struggle to predict the correct TEO graph. In the full reconstruction setting (All Pairs), GPT-5.1 correctly predicts only 3.3% of the graphs and it falls to 1.7% two other models. This is further reflected in the GSED scores: GPT-5.1 achieves the similarity of 0.61, followed by Qwen3-VL-32B (0.54) and InternVL3.5-8B (0.49), indicating substantial structural differences from the ground-truth graphs. The similarity increases in the simplified (GT Edges Only) settings, raising to 0.80 for GPT-5.1 and 0.78 for Qwen3-VL-32B. This is expected, as ignoring predictions for independent pairs removes spurious edges, leading to graphs that more closely match the ground truth (See Figure 4 in the Appendix A.3). However, this highlights the limitations of the models, as they struggle to accurately reconstruct the full temporal execution order of the plan in the simplest case, e.g. only 11.7% of exact matches for GPT-5.1. See Figures 2 and 3 in the Appendix A.3 for examples of reconstructed graphs.

6.2 Conclusion

We introduced MATEO, a publicly available multimodal benchmark designed to evaluate and advance the multimodal temporal reasoning necessary for planning of LVLMs via the Temporal Execution Order task. We provided comprehensive guidelines for extending the benchmark to new recipes, languages, and domains, enabling scalable, reproducible use for both evaluation and model development. We evaluated six open- and closed-source models and demonstrated that multimodality is critical for successfully performing the TEO task. Yet, even the best-performing model achieved modest performance, highlighting a possible explanation for the limited planning capabilities of current LVLMs. We believe MATEO can serve for future research and improvements in multimodal temporal reasoning and real-world planning.

Limitations

The main limitation of this work lies in the prompting strategies. We observed that models often interpreted instructions differently, requiring minor model-specific adjustments, such as rephrasing the output format or clarifying the reasoning instructions, and preventing a fully uniform, directly comparable evaluation. Specifically, InternVL3.5-8B failed to consistently follow the required output format unless a special <*assistant*> token was removed. We did not observe this issue with

InternVL3.5-38b. Similarly, GPT-5.1 tended to bypass the requested chain-of-thought reasoning and produce only a final answer. To fix it, we added the instruction: "you must follow the exact reasoning steps provided in the examples before providing the final answer" at the end of the prompt. Additionally, due to resource constraints, we were unable to explore multimodal in-context learning (ICL), leaving open the question of how combining image and text examples in ICL would affect performance. Finally, our experiments were conducted only in English; thus, the findings may not generalize to other languages.

Ethical Statements

Although we believe this work raises no direct ethical concerns, we cannot fully rule out potential dual-use implications of our findings.

References

James F Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Tom Bylander. 1994. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69(1-2):165–204.

Nathanael Chambers. 2013. Navytime: Event and time ordering from raw text.

Delong Chen, Theo Moutakanni, Willy Chung, Yejin Bang, Ziwei Ji, Allen Bolourchi, and Pascale Fung. 2025. Planning with reasoning using vision language world model. *arXiv preprint arXiv:2509.02722*.

Leon RA Derczynski. 2017. *Automatically ordering events and times in text*, volume 677. Springer.

Sorouralsadat Fatemi and Yuheng Hu. 2024. Enhancing financial question answering with a multi-agent reflection framework. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 530–537.

Gregor Geigle, Florian Schneider, Carolin Holtermann, Chris Biemann, Radu Timofte, Anne Lauscher, and Goran Glavaš. 2025. Centurio: On drivers of multilingual ability of large vision-language model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2831–2881, Vienna, Austria. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Naresh Gupta, Dana S Nau, and 1 others. 1991. Complexity results for blocks-world planning. In *AAAI*, volume 91, pages 629–633.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Yiyan Ji, Haoran Chen, Qiguang Chen, Chengyue Wu, Libo Qin, and Wanxiang Che. 2025. Mpcc: A novel benchmark for multimodal planning with complex constraints in multimodal large language models. In *Proceedings of the 33rd ACM International Conference on Multimedia*, MM '25, page 5188–5197, New York, NY, USA. Association for Computing Machinery.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Yash Kumar Lal, Vanya Cohen, Nathanael Chambers, Niranjan Balasubramanian, and Ray Mooney. 2024. CaT-bench: Benchmarking language model understanding of causal and temporal dependencies in plans. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 19336–19354, Miami, Florida, USA. Association for Computational Linguistics.

Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. 2024a. Llava-onevision: Easy visual task transfer. *Preprint*, arXiv:2408.03326.

Kanxue Li, Baosheng Yu, Qi Zheng, Yibing Zhan, Yuhui Zhang, Tianle Zhang, Yijun Yang, Yue Chen, Lei Sun, Qiong Cao, Li Shen, Lusong Li, Dapeng Tao, and Xiaodong He. 2024b. Muep: A multimodal benchmark for embodied planning with foundation models. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence,*

IJCAI-24, pages 129–138. International Joint Conferences on Artificial Intelligence Organization. Main Track.

Fangru Lin, Emanuele La Malfa, Valentin Hofmann, Elle Michelle Yang, Anthony G. Cohn, and Janet B. Pierrehumbert. 2024. Graph-enhanced large language models in asynchronous plan reasoning. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Yujie Lu, Pan Lu, Zhiyu Chen, Wanrong Zhu, Xin Eric Wang, and William Yang Wang. 2024. Multimodal procedural planning via dual text-image prompting. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10931–10954, Miami, Florida, USA. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Tianjun Mo, Ziyi Jiang, and Qichang Zheng. 2025. Interactive ai agent for code refactoring assistance: A study on decision-making strategies and human-agent collaboration effectiveness. *Academia Nexus Journal*, 4(1).

Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. 2014. Flow graph corpus from recipe texts. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2370–2377, Reykjavik, Iceland. European Language Resources Association (ELRA).

Liangming Pan, Jingjing Chen, Jianlong Wu, Shaoteng Liu, Chong-Wah Ngo, Min-Yen Kan, Yu-Gang Jiang, and Tat seng Chua. 2020. Multi-modal cooking workflow construction for food recipes. *Proceedings of the 28th ACM International Conference on Multimedia*.

Rebecca Passonneau. 2006. Measuring agreement on set-valued items (MASI) for semantic and pragmatic annotation. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, Genoa, Italy. European Language Resources Association (ELRA).

Lu Qiu, Yi Chen, Yuying Ge, Yixiao Ge, Ying Shan, and Xihui Liu. 2025. Egoplan-bench2: A benchmark for multimodal large language model planning in real-world scenarios. *Preprint*, arXiv:2412.04447.

Tayiba Raheem and Gahangir Hossain. 2025. Agentic ai systems: Opportunities, challenges, and trustworthiness. In *2025 IEEE International Conference on Electro Information Technology (eIT)*, pages 618–624. IEEE.

Anand S Rao, Michael P Georgeff, and 1 others. 1995. Bdi agents: From theory to practice. In *Icmas*, volume 95, pages 312–319.

Gabriel Roccabruna, Massimo Rizzoli, Giuseppe Riccardi, and 1 others. 2024. Will llms replace the encoder-only models in temporal relation classification? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Yewei Song, Saad Ezzini, Xunzhu Tang, Cedric Lothritz, Jacques Klein, Tegawende Bissyande, Andrey Boytsov, Ulrick Ble, and Anne Goujon. 2024. Enhancing text-to-sql translation for financial system design. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*, ICSE-SEIP '24, page 252–262, New York, NY, USA. Association for Computing Machinery.

Samdyuti Suri, Sankar Narayan Das, Kapil Singi, Kuntal Dey, Vibhu Saujanya Sharma, and Vikrant Kaulgud. 2023. Software engineering using autonomous agents: Are we there yet? In *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 1855–1857. IEEE.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the planning abilities of large language models: a critical investigation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Siddharth Vashishtha, Adam Poliak, Yash Kumar Lal, Benjamin Van Durme, and Aaron Steven White. 2020. Temporal reasoning in natural language inference. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4070–4078.

Weiyun Wang, Zhangwei Gao, Lixin Gu, Hengjun Pu, Long Cui, Xingguang Wei, Zhaoyang Liu, Linglin Jing, Shenglong Ye, Jie Shao, Zhaokai Wang, Zhe Chen, Hongjie Zhang, Ganlin Yang, Haomin Wang, Qi Wei, Jinhui Yin, Wenhao Li, Erfei Cui, and 56 others. 2025. Internvl3.5: Advancing open-source multimodal models in versatility, reasoning, and efficiency. *Preprint*, arXiv:2508.18265.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Joseph Weizenbaum. 1966. Eliza—a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Michael Wooldridge and Nicholas R Jennings. 1995. Intelligent agents: Theory and practice. *The knowledge engineering review*, 10(2):115–152.

Te-Lin Wu, Alex Spangher, Pegah Alipoormolabashi, Marjorie Freedman, Ralph Weischedel, and Nanyun

Peng. 2022. Understanding multimodal procedural knowledge by sequencing multimodal instructional manuals. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4525–4542, Dublin, Ireland. Association for Computational Linguistics.

Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. 2024. Travelplanner: a benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org.

Yoko Yamakata, Shinsuke Mori, and John Carroll. 2020. English recipe flow graph corpus. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 5187–5194, Marseille, France. European Language Resources Association.

Wentao Zhang, Lingxuan Zhao, Haochong Xia, Shuo Sun, Jiaze Sun, Molei Qin, Xinyi Li, Yuqing Zhao, Yilei Zhao, Xinyu Cai, and 1 others. 2024. A multimodal foundation agent for financial trading: Tool-augmented, diversified, and generalist. In *Proceedings of the 30th acm sigkdd conference on knowledge discovery and data mining*, pages 4314–4325.

Xi Zhang and Qixuan Zhu. 2023. Ai-enabled network-functions virtualization and software-defined architectures for customized statistical qos over 6g massive mimo mobile wireless networks. *IEEE Network*, 37(2):30–37.

Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. 2024. Natural plan: Benchmarking llms on natural language planning. *Preprint*, arXiv:2406.04520.

A Appendix

A.1 Experimental details

We used the HuggingFace framework to run experiments with open-source models. For generation, we employed greedy decoding (`do_sample=False`, `num_beams=1`) for all open-source models to foster reproducibility. We used the following model checkpoints: Qwen3-VL-32B[11], Qwen3-VL-8B[12], InternVL3.5-8b[13], InternVL3.5-38b[14], and LLaVA-OneVision-7b[15].

[11] https://huggingface.co/Qwen/Qwen3-VL-32B-Instruct

[12] https://huggingface.co/Qwen/Qwen3-VL-8B-Instruct

[13] https://huggingface.co/OpenGVLab/InternVL3_5-8B-HF

[14] https://huggingface.co/OpenGVLab/InternVL3_5-38B-HF

[15] https://huggingface.co/lmms-lab/llava-onevision-qwen2-7b-ov

To run the closed-source GPT-5.1[16] model, we used OpenAI Batch API [17]. As with the open-source models, we used greedy decoding for reproducibility (`temperature=0`). Since we wanted the model to follow our reasoning steps (e.g. CoT experiment), we disabled the model's default internal reasoning by setting `reasoning={"effort": "none"}`.

For fine-tuning, we used a batch size of 1, a learning rate of 1e-5, and a weight decay of 0.01. The models were fine-tuned for 1 epoch. We used Low-Rank Adaptation (LoRA) (Hu et al., 2021) to fine-tune the model. We set alpha to 32 and rank to 16. LoRA adapters are applied to the query, key, value, and output matrices of the attention layers.

Experiments involving Qwen3-VL-32B, InternVL3.5-38B, and fine-tuning were conducted using two NVIDIA A100 GPUs with 80 GiB of memory, while all other experiments were run on a single NVIDIA A100 with 80 GiB of memory. For the experiments with GPT-5.1, we spent a total of around $100, considering both experimentation and computing the results presented in this article. While for collecting the corpus, we spent a total of around £800, considering both tests and the actual annotation.

The parameters used to screen the participant sample provided by Prolific were as follows:

- First language: Italian
- Country of Birth: Italy
- Education: High school diploma/A-levels or above
- Approval rate: 95-100
- Number of previous submissions: 100-10000

A.2 Further error analysis

We analyze performance across individual TEO classes to identify the most challenging cases. Table 5 reports F1 scores for the *before*, *independent*, and *after* classes. Because the ground-truth labels for the *after* class are defined on the swapped step order, we compute F1 scores for this class on the swapped dataset. In contrast, scores for the *before* and *independent* classes are computed using the original step order. Under the baseline zero-shot setting, models exhibit a bias toward *before* and

[16] Model snapshot: gpt-5.1-2025-11-13

[17] https://platform.openai.com/docs/guides/batch

after relations, lowering performance on the *independent* class. Performance on the *independent* class improves substantially with task instructions, ICL, and CoT prompting, while gains for the *before* class are more moderate. The *after* class remains challenging across all prompting settings, with the difficulty being particularly notable under CoT prompting. Except for LLaVA-OneVision-7B, all models exhibit a drop in performance on the *after* class when using CoT. This effect is especially evident for Qwen3-VL-32B, whose F1 score drops from 0.79 with task instructions to 0.30 under CoT, corresponding to a 62% decrease in performance. A closer analysis reveals that, under CoT prompting, the model predicts *before* in 65% of cases where the ground truth is *after*, while correctly predicting *after* in only 19% of such cases. These results suggest a strong bias inherited from pre-training, where models struggle to reason under reversed temporal execution order.

Higher per-class scores compared to the consistency-based accuracy indicate that models struggle to produce consistent predictions across the original and swapped step orders. Qwen3-VL-32B performs best, producing consistent responses in 82% of cases for both *before–after* and *independent* relations under its best-performing prompting setting. Interestingly, GPT-5.1 underperforms, particularly for *before–after* relations, achieving 68% consistency for *before–after* pairs and 81% for *independent* pairs. In contrast, smaller-scale models struggle substantially, especially on *before–after* relations. For example, InternVL3.5-8B achieves only 34% consistency under its best-performing prompting setting. These inconsistencies expose fundamental limitations in the models' temporal reasoning and require caution when deploying them in planning scenarios.

A.3 Additional Material

Model	Class	Baseline	Instructions	ICL	CoT	Self-Reflect.
GPT-5.1	Before	0.8	0.83	0.81	0.8	0.82
	Independent	0.33	0.8	0.87	0.88	0.89
	After	0.64	0.76	0.74	0.73	0.75
Qwen3-VL-32B	Before	0.77	0.79	0.79	0.78	0.78
	Independent	0.07	0.62	0.88	0.84	0.78
	After	0.61	0.79	0.77	0.30	0.36
InternVL3.5-38B	Before	0.57	0.75	0.74	0.78	0.48
	Independent	0.06	0.74	0.76	0.85	0.83
	After	0.34	0.6	0.53	0.16	0.15
InternVL3.5-8B	Before	0.6	0.35	0.63	0.71	0.67
	Independent	0.0	0.73	0.63	0.74	0.69
	After	0.33	0.51	0.47	0.18	0.16
Qwen3-VL-8B	Before	0.76	0.69	0.73	0.78	0.78
	Independent	0.0	0.74	0.74	0.86	0.86
	After	0.21	0.49	0.43	0.19	0.13
LLaVA-OneVision-7B	Before	0.00	0.00	0.58	0.4	0.16
	Independent	0.00	0.00	0.06	0.35	0.05
	After	0.00	0.00	0.14	0.47	0.01

Table 5: Per-class F1 scores for the TEO task. Scores above the dashed line are computed on the original step order, and those below on the swapped order. Models in zero-shot show bias toward *before* and *after* relations, lowering *independent* performance. Prompting improves *independent* and *after* scores; however, *after* remains challenging, suggesting difficulties in reasoning over reversed temporal order.

Model	Recon. setting	GED (↓)	GSED (↑)	Exact Matches (# (%))
GPT-5.1	All Pairs	6.37 ± 4.45	0.61 ± 0.24	2 (3.3%)
	GT Edges Only	3.15 ± 2.86	0.80 ± 0.16	7 (11.7%)
Qwen3-VL-32B	All Pairs	7.47 ± 5.21	0.54 ± 0.27	1 (1.7%)
	GT Edges Only	3.47 ± 2.83	0.78 ± 0.16	6 (10.0%)
InternVL3.5-38B	All Pairs	11.87 ± 8.30	0.36 ± 0.29	0 (0%)
	GT Edges Only	5.59 ± 3.80	0.62 ± 0.21	0 (0%)
InternVL3.5-8B	All Pairs	16.23 ± 4.92	0.04 ± 0.05	0 (0%)
	GT Edges Only	16.23 ± 4.92	0.04 ± 0.05	0 (0%)
Qwen3-VL-8B	All Pairs	7.30 ± 4.91	0.55 ± 0.25	1 (1.7%)
	GT Edges Only	3.98 ± 3.88	0.76 ± 0.20	5 (8.3%)
LLaVA-OneVision-7B	All Pairs	17.98 ± 14.71	0.24 ± 0.27	0 (0%)
	GT Edges Only	5.68 ± 3.20	0.63 ± 0.19	2 (3.3%)
$\text{InternVL3.5-8B}_{\text{Fine-tuned}}$	All Pairs	8.93 ± 6.43	0.49 ± 0.26	1 (1.7%)
	GT Edges Only	5.28 ± 3.70	0.68 ± 0.20	5 (8.3%)

Table 6: Recipe-level evaluation comparing reconstructed and ground-truth graphs. The table reports Graph Edit Distance (the lower, the better), graph similarity (the higher, the better), and the number of exact graph matches (i.e. number of graphs for which all the relations have been predicted correctly). Results are shown for two reconstruction settings: using all predicted dependencies and restricting to ground-truth edges only.

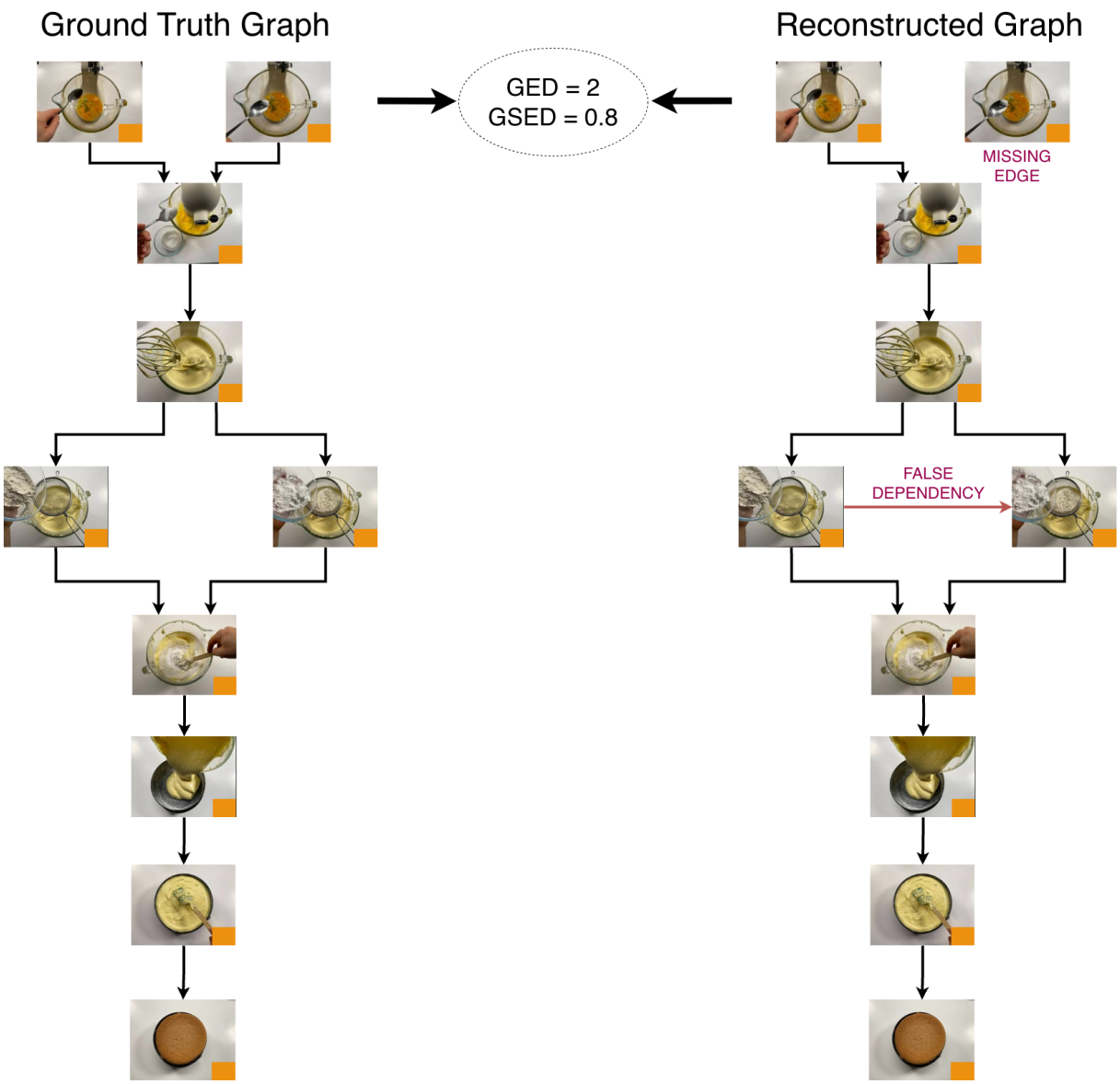


Figure 2: Example of a ground-truth (left) and reconstructed (right) graph. The Graph Edit Distance is 2, reflecting one missing and one spurious edge in the predicted graph.

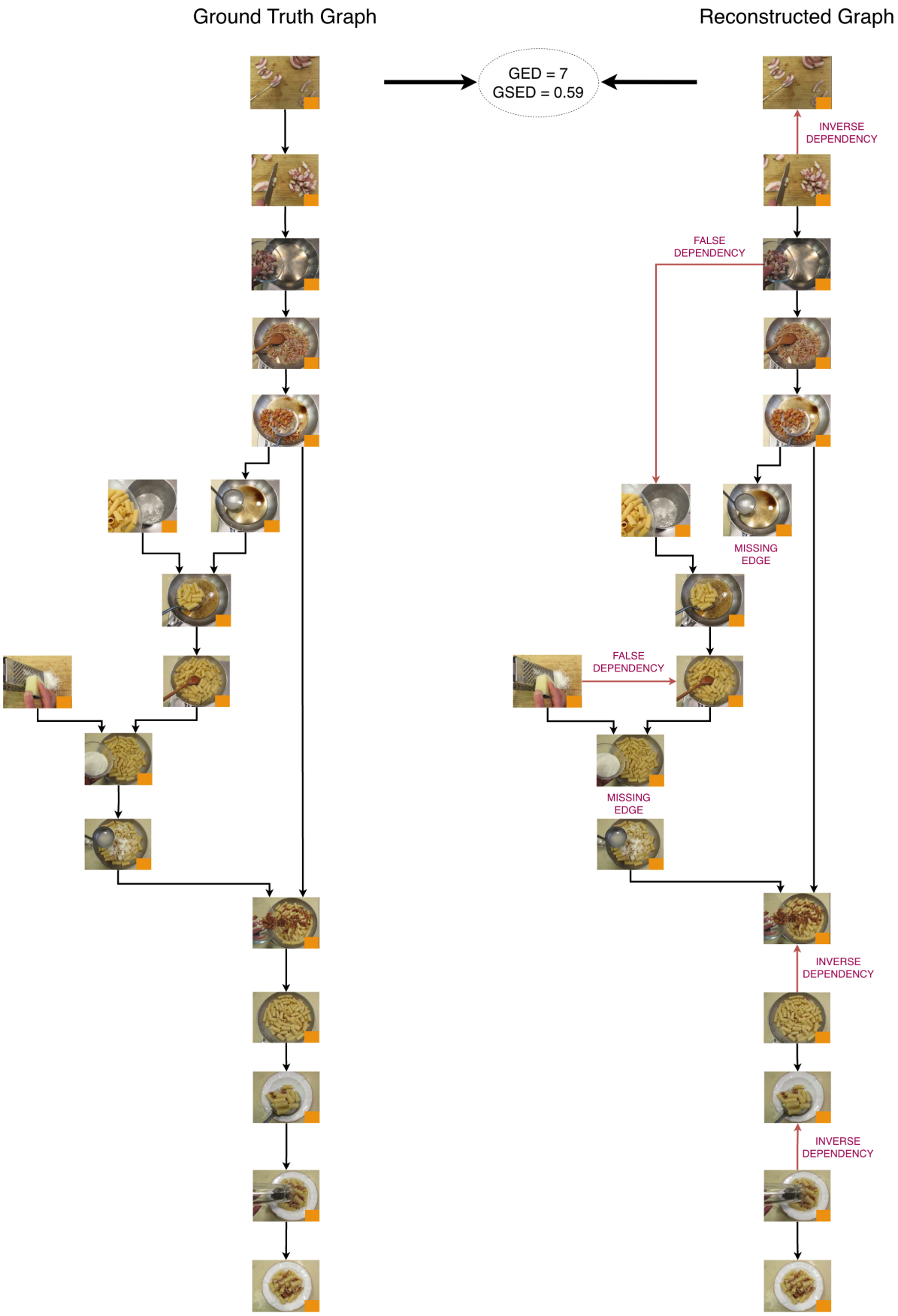


Figure 3: Example of a ground-truth (left) and reconstructed (right) graph.

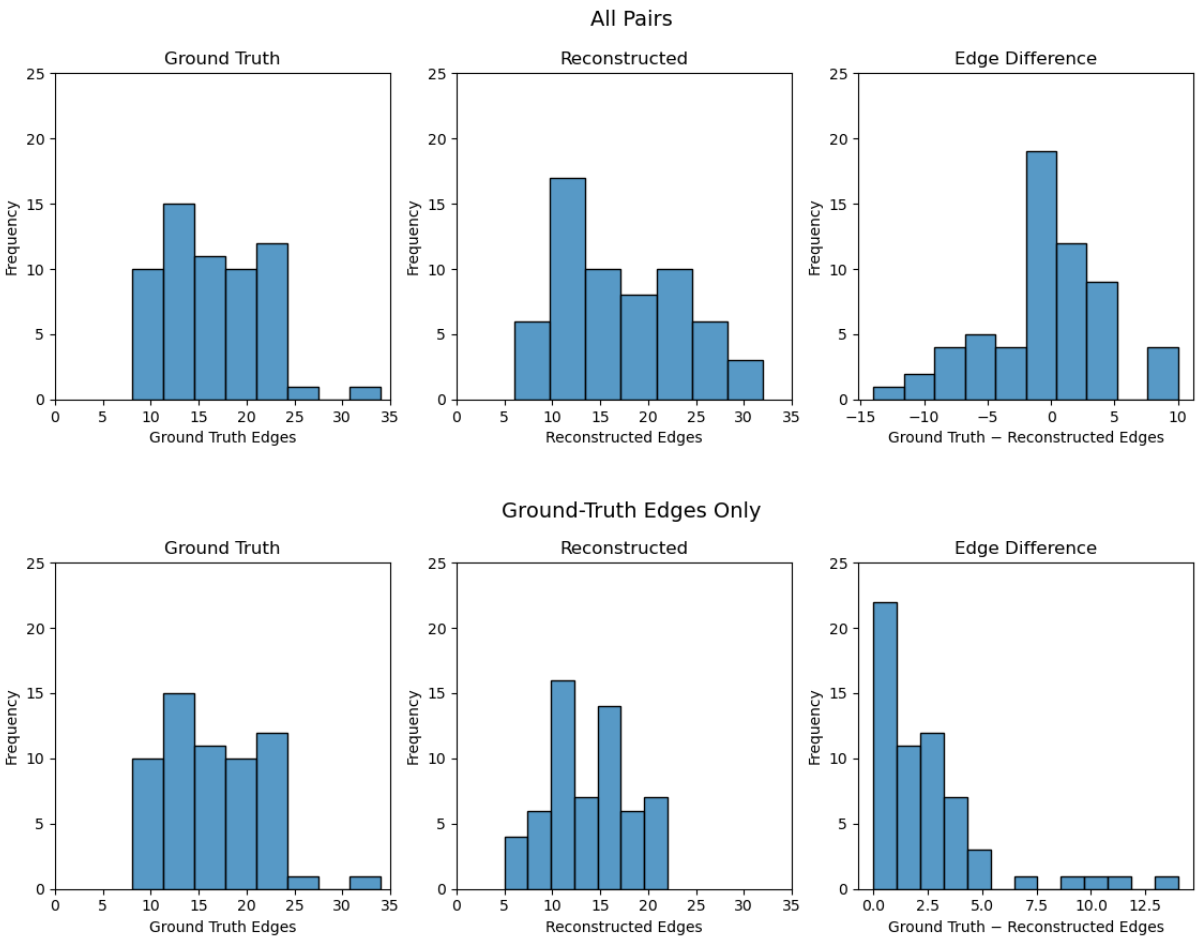


Figure 4: Distribution of the number of edges per recipe for ground-truth graphs, reconstructed graphs, and their difference. The first row shows reconstruction over all pairs, while the second row restricts reconstruction to ground-truth edges only. Negative values in the first-row difference distribution indicate spurious dependencies predicted between independent pairs, resulting in graphs with an excessive number of edges. Restricting reconstruction to ground-truth edges removes these spurious predictions, yielding graphs that more closely match the ground truth (second row).

Prompt Setting	Modality	Prompt
Baseline	Image-only	Using ONLY the information in the Context, answer the following three questions in EXACTLY this format: Q1: The answer is: <Yes/No/I don't know>. Q2: The answer is: <Yes/No/I don't know>. Q3: The answer is: <Yes/No/I don't know>. Do not add anything else. Do not explain. Do not change the format. Context: Step A picture: [image_A] Step B picture: [image_B] Questions: Q1: Must Step A be executed before Step B? Q2: Must Step A be executed after Step B? Q3: Can Step A and Step B be executed in parallel?
	Text-only	Using ONLY the information in the Context ... Context: Step A description: [description_A] Step B description: [description_B] Questions ...
	Image-Text	Using ONLY the information in the Context ... Context: Step A picture: [image_A] Step A description: [description_A] Step B picture: [image_B] Step B description: [description_B] Questions ...
Instructions	Image-only	Your task is to determine the dependency order between two steps in a recipe. Follow these rules: - **Before**: Step A must be executed before Step B if the outcome of Step A is required to complete Step B (i.e., Step B depends on Step A). - **After**: Step A must be executed after Step B if the outcome of Step B is required to complete Step A (i.e., Step A depends on Step B). -**Parallel**: Step A and Step B can be executed in parallel if neither step depends on the outcome of the other; therefore, their order of execution can be arbitrary. Using ONLY the information in the Context ... Context: ... Questions: ...
	Text-only	Your task is to determine the dependency order between two steps in a recipe. Follow these rules: - Before: ... - After: ... - Parallel: ... Ignore sequencing terms (e.g., 'first', 'then', 'lastly', and other words that may appear in the text for the natural flow of the recipe) when determining the execution order, and focus only on the action itself.

Prompt Setting	Modality	Prompt
		Using ONLY the information in the Context ... Context: ... Questions: ...
	Image-Text	Your task is to determine the dependency order between two steps in a recipe. Follow these rules: - Before: ... - After: ... - Parallel: ... Ignore sequencing terms (e.g., 'first', 'then', 'lastly', and other words that may appear in the text for the natural flow of the recipe) when determining the execution order, and focus only on the action itself. Also note that the text description may include partial or full references to steps not shown in the image; in such cases, rely on the actions depicted in the image. Using ONLY the information in the Context ... Context: ... Questions: ...
In-context learning	Image-only	Your task is to determine the dependency order between two steps in a recipe. Follow these rules: - Before: ... - After: ... - Parallel: ... **You will be shown three examples demonstrating how to solve the task using text-based step descriptions. However, your actual input will consist of images, and your reasoning should be based on the actions depicted in those images.** Examples: Step A description: Grate the lemon zest. Step B description: Put the grated lemon zest into the strawberry sauce. Questions: Q1: Must Step A be executed before Step B? Q2: Must Step A be executed after Step B? Q3: Can Step A and Step B be executed in parallel? Q1: The answer is: Yes. Q2: The answer is: No. Q3: The answer is: No. Explanation: Step B explicitly depends on Step A - lemon zest must already be grated (Step A) before it can be put into the strawberry sauce (Step B); therefore, Step A must be executed before Step B. Step A description: Add the celery. Step B description: Then add carrots. Questions: Q1: Must Step A be executed before Step B? Q2: Must Step A be executed after Step B? Q3: Can Step A and Step B be executed in parallel? Q1: The answer is: No.

Prompt Setting	Modality	Prompt
		Q2: The answer is: No. Q3: The answer is: Yes. Explanation: Both actions are independent; neither step produces something the other one requires. Step A description: Pour it into the cup. Step B description: Measure out raspberry juice. Questions: Q1: Must Step A be executed before Step B? Q2: Must Step A be executed after Step B? Q3: Can Step A and Step B be executed in parallel? Q1: The answer is: No. Q2: The answer is: Yes. Q3: The answer is: No. Explanation: Step A relies on the outcome of Step B - raspberry juice must be poured into the cup (Step A) after it is measured out (Step B); therefore, Step A must be executed after Step B. Using ONLY the information in the Context ... Context: ... Questions: ...
	Text-only	Your task is to determine the dependency order between two steps in a recipe. Follow these rules: - Before: ... - After: ... - Parallel: ... Ignore sequencing terms ... **You will be shown three examples demonstrating how to solve the task using text-based step descriptions.** Examples: ... Using ONLY the information in the Context ... Context: ... Questions: ...
	Image-Text	Your task is to determine the dependency order between two steps in a recipe. Follow these rules: - Before: ... - After: ... - Parallel: ... Ignore sequencing terms ... **You will be shown three examples demonstrating how to solve the task using text-based step descriptions. However, your actual input will consist of both images and text descriptions, and your reasoning should be based on both actions shown in the images and the accompanying textual descriptions.** Examples: ... Using ONLY the information in the Context ... Context: ... Questions: ...
CoT	Image-only	Your task is to determine the dependency order between two steps in a recipe. **You must choose from: Before, After, or Parallel**. Follow these rules:

Prompt Setting	Modality	Prompt
		- Before: ... - After: ... - Parallel: ... **You will be shown three examples demonstrating how to solve the task using text-based step descriptions. However, your actual input will consist of images, and your reasoning should be based on the actions depicted in those images. You must follow the reasoning steps shown in the examples before answering.** Examples: Step A description: Grate the lemon zest. Step B description: Put the grated lemon zest into the strawberry sauce. Step A produces: Grated lemon zest. Step B produces: Lemon zest inside the strawberry sauce. Step A requires: A lemon. Step B requires: Lemon zest that has been grated. Dependency analysis: Step B explicitly depends on Step A - lemon zest must already be grated (Step A) before it can be put into the strawberry sauce (Step B); therefore, Step A must be executed before Step B. The answer is: Before. Step A description: Add the celery. Step B description: Then add carrots. Step A produces: A component with the celery added. Step B produces: A component with the carrots added. Step A requires: The celery. Step B requires: The carrots. Dependency analysis: Each step adds a separate ingredient, and neither depends on the other, so they can occur in any order. The answer is: Parallel. Step A description: Pour it into the cup. Step B description: Measure out raspberry juice. Step A produces: The cup with raspberry juice poured in it. Step B produces: Raspberry juice that was measured out. Step A requires: Raspberry juice that was measured out. Step B requires: Raspberry juice. Dependency analysis: Step A relies on the outcome of Step B - raspberry juice must be poured into the cup (Step A) after it is measured out (Step B); therefore, Step A must be executed after Step B. The answer is: After. Step A picture: ...
	Text-only	Your task is to determine the dependency order between two steps in a recipe. **You must choose from: Before, After, or Parallel**. Follow these rules:

Prompt Setting	Modality	Prompt
		- Before: ... - After: ... - Parallel: ... Ignore sequencing terms ... **You will be shown three examples demonstrating how to solve the task using text-based step descriptions. You must follow the reasoning steps shown in the examples before answering.** Examples: ... Step A description: ...
	Image-Text	Your task is to determine the dependency order between two steps in a recipe. **You must choose from: Before, After, or Parallel**. Follow these rules: - Before: ... - After: ... - Parallel: ... **You will be shown three examples demonstrating how to solve the task using text-based step descriptions. However, your actual input will consist of both images and text descriptions, and your reasoning should be based on both actions shown in the images and the accompanying textual descriptions. You must follow the reasoning steps shown in the examples before answering.** Examples: ... Step A picture: ... Step A description: ...
Self-reflection	Image-only	Your task is to determine the dependency order between two steps in a recipe. **You must choose from: Before, After, or Parallel**. Follow these rules: - Before: ... - After: ... - Parallel: ... You will be shown three examples demonstrating how to solve the task using text-based step descriptions. However, your actual input will consist of images, and your reasoning should be based on the actions depicted in those images. You must follow the reasoning steps shown in the examples before answering. **After you answer the question, review your reasoning and check whether your answer logically follows from the context and dependencies you identified. After self-reflection, provide your final answer, confirming or correcting your initial choice.** Examples: Step A description: Grate the lemon zest. Step B description: Put the grated lemon zest into the strawberry sauce.

Prompt Setting	Modality	Prompt
		Step A produces: Grated lemon zest. Step B produces: Lemon zest inside the strawberry sauce. Step A requires: A lemon. Step B requires: Lemon zest that has been grated. Dependency analysis: Step B explicitly depends on Step A - lemon zest must already be grated (Step A) before it can be put into the strawberry sauce (Step B); therefore, Step A must be executed before Step B. The answer is: Before. **Reflection: <your_reflection>** **The final answer: <final_answer>** ... Step A picture: ...
	Text-only	Your task is to determine the dependency order between two steps in a recipe. **You must choose from: Before, After, or Parallel**. Follow these rules: - Before: ... - After: ... - Parallel: ... Ignore sequencing terms ... You will be shown three examples demonstrating how to solve the task using text-based step descriptions. You must follow the reasoning steps shown in the examples before answering. **After you answer the question, review your reasoning and check whether your answer logically follows from the context and dependencies you identified. After self-reflection, provide your final answer, confirming or correcting your initial choice.** Examples: ... Step A description: ...
	Image-Text	Your task is to determine the dependency order between two steps in a recipe. **You must choose from: Before, After, or Parallel**. Follow these rules: - Before: ... - After: ... - Parallel: ... **You will be shown three examples demonstrating how to solve the task using text-based step descriptions. However, your actual input will consist of both images and text descriptions, and your reasoning should be based on both actions shown in the images and the accompanying textual descriptions. You must follow the reasoning steps shown in the examples before answering.** Examples: ... Step A picture: ...

Prompt Setting	Modality	Prompt
		Step A description: ...

Table 7: Prompt schema used for models evaluation under image-only, text-only, and image–text modalities. Repeated or previously described components are omitted for brevity.

Modality	Prompt
Image-only	<User> Your task is to determine the dependency order between two steps in a recipe. Follow these rules: - **Before**: Step A must be executed before Step B if the outcome of Step A is required to complete Step B (i.e., Step B depends on Step A). - **After**: Step A must be executed after Step B if the outcome of Step B is required to complete Step A (i.e., Step A depends on Step B). -**Parallel**: Step A and Step B can be executed in parallel if neither step depends on the outcome of the other; therefore, their order of execution can be arbitrary. Step A picture: [image_A] Step B picture: [image_B] Questions: Q1: Must Step A be executed before Step B? Q2: Must Step A be executed after Step B? Q3: Can Step A and Step B be executed in parallel? <Assistant> Q1: The answer is: Yes. Q2: The answer is: No. Q3: The answer is: No.
Text-only	<User> Your task is to determine the dependency order between two steps in a recipe. Follow these rules: - Before: ... Ignore sequencing terms (e.g., 'first', 'then', 'lastly', and other words that may appear in the text for the natural flow of the recipe) when determining the execution order, and focus only on the action itself. Step A description: [description_A] Step B description: [description_B] Questions: Q1: Must Step A be executed before Step B? Q2: Must Step A be executed after Step B? Q3: Can Step A and Step B be executed in parallel? <Assistant> Q1: The answer is: No. Q2: The answer is: No. Q3: The answer is: Yes.
Image-Text	<User> Your task is to determine the dependency order between two steps in a recipe. Follow these rules: - Before: ... Ignore sequencing terms (e.g., 'first', 'then', 'lastly', and other words that may appear in the text for the natural flow of the recipe) when determining the execution order, and focus only on the action itself. Also note that the text description may include partial or full references to steps not shown in the image; in such cases, rely on the actions depicted in the image. Step A description: [description_A] Step B description: [description_B] Questions: Q1: Must Step A be executed before Step B? Q2: Must Step A be executed after Step B? Q3: Can Step A and Step B be executed in parallel? <Assistant>

Q1: The answer is: No.
Q2: The answer is: Yes.
Q3: The answer is: No.

Table 8: Example of prompt–response pairs used for fine-tuning across image-only, text-only, and image–text modalities. Each example includes instructions, input structure, and expected binary outputs. The <User> and <Assistant> tokens are model-specific and are included here solely for illustrative purposes.

1. Purpose

Welcome! The goal of our research project is to test whether humans can recognize the order of steps in a recipe. For this, we need annotated data, so we need your help!

Note

The assignment will be divided into two parts. The first part includes a test (***Comprehension Check***) to verify your understanding of the assignment. If you pass, you'll move on to the second part, where you'll have to write down a series of recipes.

In the test part you will have to write down 2 recipes and in the annotation part you will have to write down 5 recipes.

2. Task

In this study, we'll ask you to note **the order of dependencies between the steps of a recipe** . We'll provide you with all the steps of a recipe along with the corresponding images showing: the action , the text description, and the step number that reflects the order of the original recipe. Note that the step number is only used to follow the flow of the recipe and may ***not*** correlate with the order of dependencies.

For each step of a recipe:

- **Find the part of the description related to the image:** first, you'll need to read the description and find the part of the text that best describes or relates to the action depicted in the image. ***Use this part of the description to decide the order of dependencies!*** The rest of the text will help you understand the flow of the recipe and should therefore be used as context.
- **Otherwise, rely on the image:** in some cases, the description of the action may not be present in the text. In these cases, we ask you to rely on the action depicted in the image, or what the image communicates.

For clarity, in the examples below, we've removed the passage description from the image, but you'll see it displayed in the annotation activity.

Figure 5: This figure shows the first two points of the annotation guidelines, that present the purpose and the general description of the task.

A.4 AI tool usage

During the development phase, we used ChatGPT and Gemini to identify and resolve implementation bugs. We also leveraged these models and Grammarly to suggest rephrasing that improves the fluency and readability of the manuscript. Additionally, we have experimented with using Google Scholar Labs to expand and refine our literature review.

A.5 Licences of the corpus and models used

All models used in our experiments have been utilised in accordance with their respective licenses. We plan to license the corpus with the Creative Commons 4.0 CC BY-NC-SA 4.0 (Attribution-NonCommercial-ShareAlike). The intended use of this corpus will be only for scientific research purposes.

3. Order of dependency

To identify the dependency order **watch the video** below and **read the descriptions** 1) dependency order, 2) concurrent execution and 3) transitivity in green and the related examples in blue .

How to identify the dependency order?

1) Order of Dependence

Given two steps J and P , P is required by J (P → J) if step P needs to be completed before step J . In other words , **the result of step P is absolutely necessary for step J to be completed** . You can think of steps as processes. A process needs some requirements to begin with. If those requirements are produced by another process, then there is a dependency between the two. To help understand better , you can refer to Example 1.1.

Example 1.1

Given the steps:

11 Grate the lemon zest;

12 Add the grated lemon zest to the strawberry sauce;

12 depends on 11 . This is because without grating the lemon zest, it is not possible to add the grated zest to the strawberries.

Figure 6: This section of the annotation guidelines offers annotators the option to watch a demonstration video that walks through the task and explains how to identify dependencies between steps. Following the video, annotators are provided with a textual description that formally defines each dependency, accompanied by illustrative examples.

Example 1.1

Given the steps:

11 Grate the lemon zest;

12 Add the grated lemon zest to the strawberry sauce;

12 depends on 11. This is because without grating the lemon zest, it is not possible to add the grated zest to the strawberries.

2) Simultaneous execution

There may be steps that have no dependencies between them, meaning they can theoretically be executed simultaneously. That is, the order in which they are executed does not affect the correct execution of the recipe. To help you understand, follow Examples 2.1, 2.2, and 2.3.

Example 2.1

Given the steps:

1 Fry a clove of garlic with a drizzle of extra virgin olive oil;

2 Add the celery;

3 Add carrots;

4 Add onions;

Step 1 must be done first. This is because the garlic must be sautéed before proceeding to the next steps. Steps 2, 3, and 4 can be done simultaneously. That is 1 → 2, 1 → 3, and 1 → 4, as shown in the figure . This is because I don't need onions to correctly perform the "add carrots " action , or vice versa . Furthermore, the recipe doesn't explicitly specify any order, so we can be sure that there is no order dependency between these steps.

Figure 7: This figure continues from the previous one, illustrating a case in which a step depends on another (Example 1.1), and presenting the formal definition of independent steps

2) Simultaneous execution

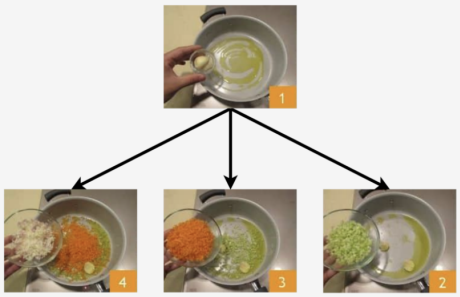
There may be steps that have no dependencies between them, meaning they can theoretically be executed simultaneously. That is, the order in which they are executed does not affect the correct execution of the recipe. To help you understand, follow Examples 2.1, 2.2, and 2.3.

Example 2.1

Given the steps:

1 Fry a clove of garlic with a drizzle of extra virgin olive oil;
2 Add the celery;
3 Add carrots;
4 Add onions;

Step 1 must be done first. This is because the garlic must be sautéed before proceeding to the next steps. Steps 2, 3, and 4 can be done simultaneously. That is 1 → 2, 1 → 3, and 1 → 4, as shown in the figure . This is because I don't need onions to correctly perform the "add carrots " action , or vice versa . Furthermore, the recipe doesn't explicitly specify any order, so we can be sure that there is no order dependency between these steps.



Example 2.2

The part that follows Example 2.1 is that steps that can ideally be performed simultaneously must finish before step 5. So it follows that 5 **depends on** 2, 3, and 4. The reason is that you can't start with step 5 without having the celery, carrots, and onions in the pan.

Figure 8: This figure shows an example given to illustrate the case in which steps are independent.

Example 2.3

Given the following steps:

1 Add ice to a glass;
2 Measure out the lemon juice;
3 Pour it;
4 Measure out the grape juice;
5 Pour it;
6 Measure and pour in the raspberry juice;
7 Mix well;

We can note that steps 2 and 3, and 4 and 5 are the breakdown of step 6. Furthermore, considering that in step 7 everything is mixed, that the recipe does not specify the order of additions, and that, for example, I do not need the lemon juice to physically pour the grape juice, we can say with good certainty that the pairs 2 - 3, 4 - 5 and step 6 can be performed simultaneously.

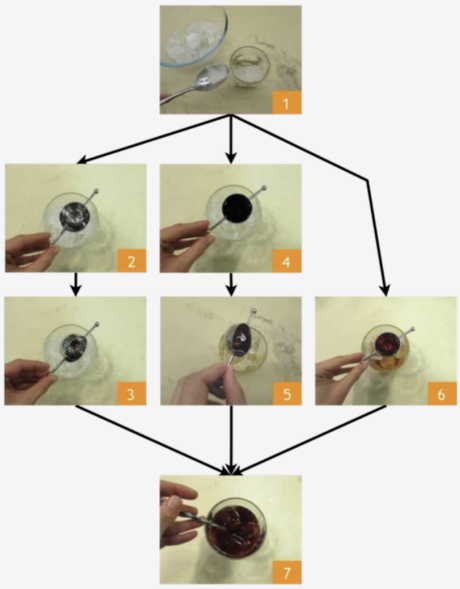
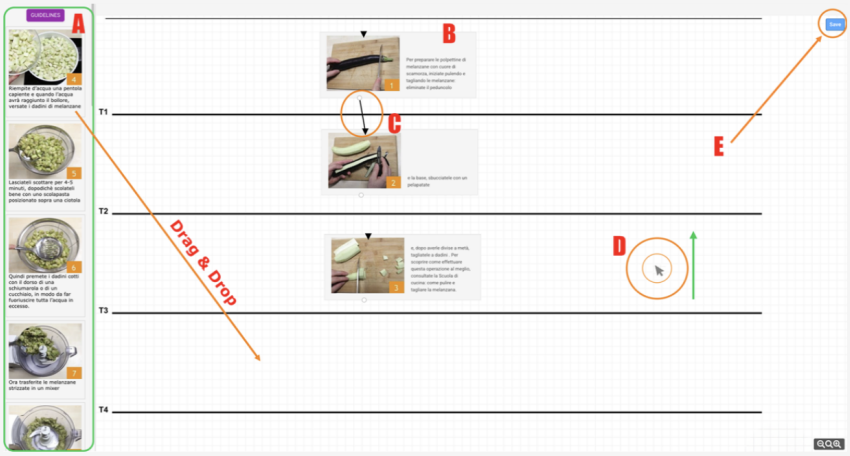


Figure 9: Example illustrating a fully annotated DAG for a recipe, with all relations clearly explained.

5. Annotation tool

You should now have a clear understanding of the task. So, let's briefly review the annotation tool. The annotation tool is illustrated in the following image:



- **A:** This is the panel with the list of recipe steps you can add to the timeline. At the top of this panel, you'll find a "Guidelines" button. This button takes you back to the guidelines you're currently reading, so you can refer back to them at any time during your annotation (for example, if you're unsure or forget something).
- **B:** To drag steps onto the timeline, you need to hold down the left mouse button on the gray area of the step.
- **C:** **To connect** two steps, click the white dot of the first step and then the black triangle of the second step. This will create a line showing the dependency between the two steps. **To delete** a connection, simply click the connection line you want to delete and press "delete" or, on the keyboard, press *DEL, DEL, or on a MAC, Cmd+Delete/Backspace* .
- **I:** To scroll down the timeline, hold down the left mouse button on any part of the timeline and move the mouse up. (*It's like scrolling a web page on a smartphone.*) The same goes for moving left and right.
- **E:** once you have connected all the steps, click on the "Save" button *to* proceed to the next recipe or finish the activity, in case there are no more recipes.
  - **Note:** There may be some rare cases where a step can't be linked to anything. In this case, once you've finished linking all the other steps, click "Save," and we'll then ask you to provide the reason why this step couldn't be linked to the others.

Figure 10: Illustration of the annotation tool as presented in the annotation guidelines. Highlighted regions (marked in red letters) indicate key functionalities, including dragging steps, adding or deleting connections, and saving the annotations.

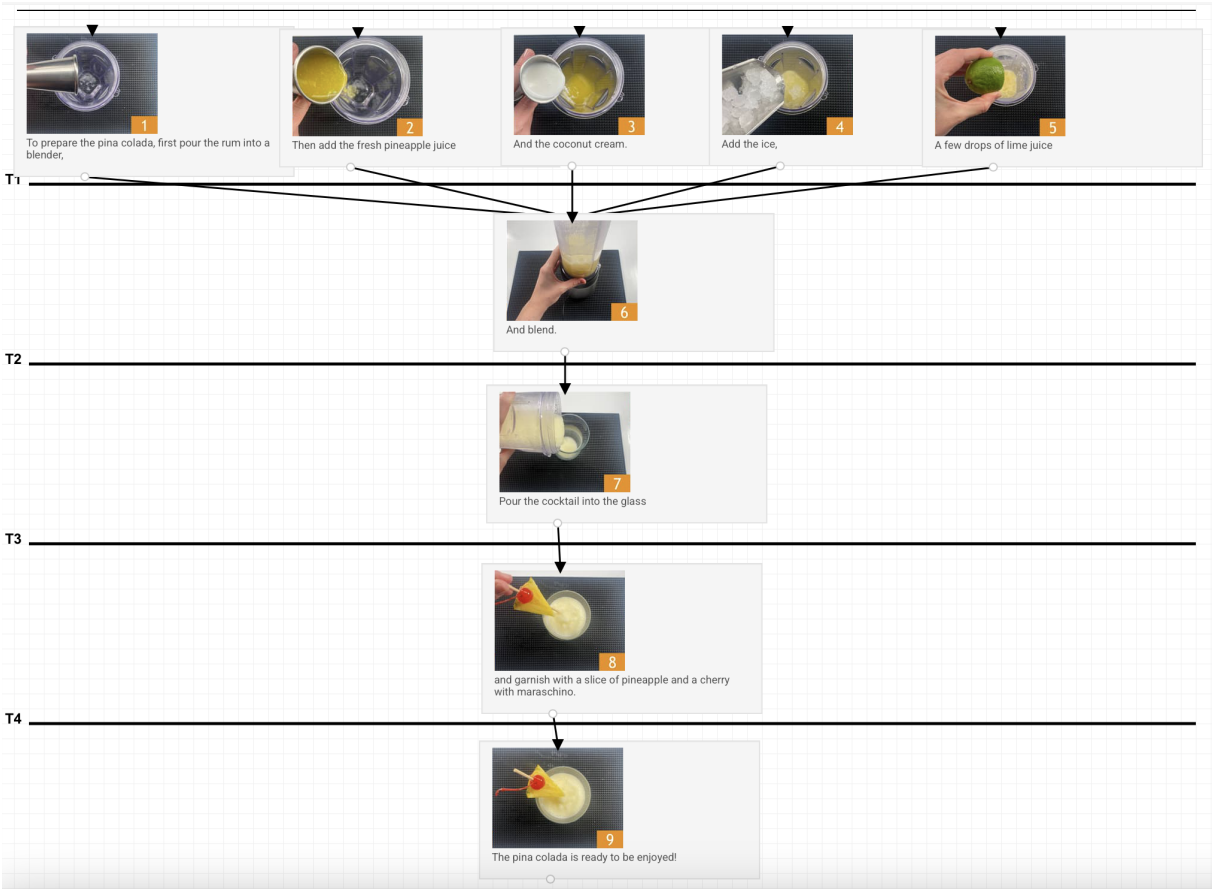


Figure 11: Example of a DAG annotated by a crowd-worker. Steps 1, 2, 3, 4, and 5 can be executed in any order (or simultaneously), thus there is no directed edge between them.