UNIVERSITÀ DI TRENTO
Department of
Information Engineering and Computer Science

Doctoral Programme in
Information Engineering and Computer Science

LANGUAGE-GROUNDED POST-COMPLETION MISTAKE DETECTION IN PROCEDURAL VIDEOS

Olga Loginova

Advisor
Prof. Andrea Passerini
Università di Trento

Co-Advisor
Prof. Elisa Ricci
Università di Trento

Co-Advisor
Prof. Jacopo Staiano
Università di Trento

January 2026

Acknowledgements

My deepest gratitude goes to my supervisors and mentors, whose guidance, standards, and generosity have profoundly shaped both this thesis and the way I think about research. I am thankful to **Prof. Andrea Passerini** for creating an environment where it was possible to pursue research with both intellectual freedom and a sense of stability, to **Prof. Elisa Ricci** for her constructive feedback on my research proposal and her expert advice on Computer Vision and beyond, to **Prof. Frank Keller** for welcoming me into his team at the University of Edinburgh and supporting my research with thoughtful guidance and insightful advice, and to **Prof. Jacopo Staiano** for kindly agreeing to serve on my advisory committee. Each of them, in a different way, set a high bar and contributed to shaping me into a more thoughtful, responsible, ethical, and grateful researcher.

I also thank the **Amazon Alexa team**, whose grant administered through Raffaella Bernardi supported my PhD journey, including conference travel and research internships.

My thanks also extend to colleagues, co-authors, annotators, and to all those who supported me with feedback, ideas, and generosity in sharing their knowledge and experience. In particular, I would like to thank **Oleksandr Bezrukov** from Gran Sasso Science Institute and **Alexey Kravets** from the University of Bath; **Gautier Dagan** and **Anil Batra** from the University of Edinburgh, and **Ravi Shekhar** from the University of Essex for inspiring discussions and collaboration on models and ideas; **Sofia Ortega Loguinova**, **Anna Loguinova**, **Vasili Noè**, **Elizaveta Loginova**, **Maya Udaka**, and **Cristina Crippa** for their efficiency, language expertise, and high-quality annotation work.

I am equally grateful to the Edinburgh group: **Irina Saparina**, **Alex Gurung**, **Ashutosh Adhikari**, **Sophie Fischer**, **Iñigo Alonso**, **Ulrich Germann**, **Miao Li**, and **Ekaterina Taktasheva**, for creating a warm atmosphere during my internship in Scotland. Their kindness, shared stories, and encouragement helped me stay grounded, especially in the final days and hours before conference deadlines.

Last but certainly not least, my deepest thanks go to my partner, **Angelo Noè**, for his patience, care, and unwavering support throughout this PhD adventure, from trouble and strife to poster design and proofreading camera-ready versions.

Abstract

Mistake detection in procedural videos is the task of identifying errors in activities such as cooking, assembly, or repair. The domain represents a critical yet underexplored challenge. This thesis focuses on Post-Completion Mistake Detection (PCMD), where a model must verify a full procedure execution and localize deviations from the intended protocol. PCMD is under-researched and still held back by fragmented error taxonomies, staged and scarce datasets, and complex, computationally demanding, often domain-specific vision-first models.

This thesis develops a unified, language-centered PCMD framework. First, it establishes the limitations of end-to-end Vision-Language Models (VLMs) for procedural verification. Through gaps in temporal reasoning of ongoing and completed actions, failures in understanding of cause-effect relations in procedural structures, and model tendencies towards "blind guessing", the thesis demonstrates that VLMs struggle with fine-grained temporal logic. The diagnostics prove that reliable mistake detection requires structured and interpretable mechanisms over black-box VLM reasoning alone. Second, to address the data bottleneck, the thesis introduces PIE-V, a semi-synthetic pipeline for generating mistake-aware datasets. Using psychology-informed error planning, PIE-V injects semantic mistakes into clean procedures. It delivers controllable, error-rich variants that approximate real-world error scenarios, in contrast to the staged mistakes of the current mistake-aware video datasets, and outperforms freeform LLM-based generation in coherence and perceived realism. Third, the thesis presents a lightweight, language-grounded PCMD framework, `ChronoFix`. The method grounds video executions into step sequences, compares raw step descriptions, semantic role representations, and action–object abstractions, and verifies the resulting traces with a Hidden Markov Model. Across CaptainCook4D, EgoPER, EgoOops, and auxiliary Assembly101 experiments, the results show that semantic-role normalization improves robustness to noisy VLM grounding and that explicit sequence modeling supports interpretable cross-dataset mistake detection.

This work advances the state of the art by (1) providing diagnostic evidence of VLM failures in temporal logic, (2) introducing a scalable pipeline for generating realistic mistakes, and (3) presenting an efficient, structure-first baseline for post-completion mistake detection.

Keywords

procedural video understanding; post-completion mistake detection; video language models; mistake-aware datasets; temporal reasoning

Contents

List of Tables

List of Figures

Signor Sinsone affrettati combatti
Vieni da me con gli strumenti adatti
Cambia i collegamenti designatti
Ottomilaseicentodiciassatti
Fai la riparazione. Tante gratti.

— Primo Levi, *Il Versificatore*

Chapter 1

Introduction

Picture this: on a Tuesday evening, a home cook makes zoodles, only to discover at the dinner table that the pasta has been boiled a few minutes too long. The texture is ruined. On a factory line, a gearbox looks finished until inspection reveals a single flipped plate. The entire assembly fails. In both cases, the task appears complete; the error emerges only after the fact.

AI systems that collaborate with people in procedural activities, such as cooking, assembly, or repair, must account for the fact that failures are common and consequential [16, 123]. In this thesis, we focus on **Post-Completion Mistake Detection (PCMD)**, the type of video-based mistake detection in procedures when the task appears complete. PCMD supports troubleshooting, retrospective quality assurance, and automated debriefing in monitored environments [40, 89], assuming access to the complete recording of the task execution.

Despite extensive progress in temporal segmentation [27], action recognition, and step prediction for correctly executed procedures [28, 57], PCMD remains underexplored [6, 79]. Existing methods are few, often dataset-specific, computationally heavy, and fragile under open-set conditions. Benchmarks also lack a unifying structure: some provide only binary error flags, while others offer fine-grained taxonomies tied to single domains, hindering generalization and interpretability [60].

Against this backdrop, we introduce mistake detection in procedural videos and position it at the intersection of Computer Vision and Natural Language Processing, with an emphasis on temporal and structured reasoning about multi-step activities.

1.1 The Context

1.1.1 Towards egocentric procedural AI assistants

Procedural video understanding lies at the intersection of Computer Vision and Natural Language Processing: from the vision side, models must parse long, fine-grained interactions with tools, objects, and changing states; from the language side, they must map observations to step descriptions, protocols, and task constraints. The broader motivation shifts from passive understanding to active assistance in first-person settings. Assistant scenarios routinely couple procedure understanding with interactive needs such as question answering, step guidance, and execution verification [63]. In these settings, procedure-following errors are not edge cases: they drive safety risks, quality failures, and costly rework, which makes verification and diagnosis a core assistant capability, not an optional add-on [6].

1.1.2 Verification vs. anticipation

While much of the work on anomaly prediction and mistake detection targets *action anticipation* (forecasting deviations early, for timely intervention) [57, 99], this thesis addresses *verification*: judging what has happened in a procedure and whether it matches the intended protocol. Procedures are goal-directed and temporally organized: steps constrain what can happen next and which intermediate states must be reached. A system can recognize many local actions and still miss that the overall protocol was violated (e.g., a required step was omitted, two steps were transposed, or a critical action was performed on the wrong object). Full-trajectory access also matters for *corrections*: recovery actions and compensations are often identifiable only once the procedure unfolds to the end [6, 123]. Modern benchmarks increasingly reflect this need via paired video–text representations, where step sequences form the reference protocol and the video is evidence for or against it [47, 96].

1.2 The Problem Definition and Scope

We define PCMD as follows: given a full video of a multi-step activity that appears complete and a reference instruction, the model must decide whether the task was executed correctly and localize any mistakes that occurred. Specifically, the task is to (i) decide whether the execution conforms to the protocol, (ii) localize segments or steps that deviate, and (iii) characterize the deviation when possible.

1.2.1 Terminology and Task Positioning

PCMD concerns identifying deviations from an expected behavior protocol in multi-step task executions. The area commonly distinguishes settings that differ in the available evidence. In *mistake recognition*, errors are diagnosed after the relevant actions have unfolded, using the full temporal context; in *mistake detection*, the system additionally localizes error segments instead of assuming pre-segmented clips; in *early/online mistake recognition*, the task moves into an anticipatory regime, where an error is flagged before it fully manifests or while only a prefix of the execution is available [6, 34]. These settings differ not only in timing, but also in what counts as sufficient evidence.

In PCMD, the model observes a complete procedure execution from start to finish and verifies it against the protocol. This supports retrospective analysis of global dependencies (e.g., detecting that a step skipped early on was never compensated later) and aligns PCMD with offline quality assurance. Thus, PCMD is also the natural setting for analyzing *corrections*: certain recovery patterns become visible only when the full trajectory is available [6, 123].

1.2.2 Challenges in Post-Completion Mistake Detection

Taxonomy and comparability issues

Procedural mistake benchmarks span heterogeneous domains (cooking, assembly, repair, chemical experiments), but datasets rarely share a compatible labeling scheme. Some emphasize structural deviations (skips, swaps), while others foreground executional issues (technique, timing), and the boundary between "acceptable variation" and true error is often unclear. Even when datasets share intuitive categories (e.g., omissions, order errors, wrong execution), they differ in granularity and interpretation, especially for executional mistakes [6]. This blocks direct cross-domain comparison and complicates transfer across domains.

Another source of fragmentation is that many taxonomies do not tie an error to *what exactly* is violated in the instruction: a wrong action, object, instrument, manner, or outcome. Coarse categories may say that something went wrong, but remain uninformative about which component of the step was violated, which limits both comparability and actionable feedback [67]. These issues motivate instruction-grounded error semantics, where mistakes are attributed to violated semantic roles, alongside a shared high-level view of error structure.

Dataset limitations and benchmarking challenges

Mistake-aware procedural datasets remain scarce, and many contain scripted mistakes [96] performed in a staged manner that can miss the cognitive subtlety of real slips and recoveries [123]. In addition, existing mistake datasets tend to be small and exhibit limited semantic and visual variance, which weakens benchmarking under real-world diversity [67]. The largest datasets, such as Assembly101 [62] and CaptainCook4D [96], still include fewer than 400 videos from a single perspective.

Staged error collection can also introduce visual bias: mistake instances may look systematically different from correct ones, pushing the benchmark away from the subtlety expected in natural settings [67]. Finally, even where mistake explanations exist, they are often free-form and not structurally aligned to instruction components, and fine-grained grounding of error timing is frequently missing, which complicates both automation and evaluation [67]. These limitations create common failure points for training and evaluation: models can pick up dataset artifacts instead of robust mistake reasoning.

Limitations of vision-first and domain-specific pipelines

Many existing methods are vision-first and heavily tailored to a dataset's annotation format. They can be computationally heavy and hard to interpret: when a model flags an error, it may not be clear whether failure stems from perception, temporal reasoning, protocol mismatch, or evaluation artifacts. Dense supervision is also suboptimal: a common pattern is end-to-end optimization on action/mistake labels that requires segment-level annotations of both correct and erroneous executions.

A further limitation is that purely motion- or execution-centric modeling can miss mistakes that primarily manifest in *outcomes*: unintended object states or incorrect spatial arrangements that look locally plausible while unfolding, but are wrong when judged by the produced effect [46]. This pushes mistake detection toward representations that track state changes and effects (not only action appearance), and toward evaluation under open-set deviations, where unseen error modes are the norm, not the exception [6, 46]. These constraints make structured, interpretable, and computationally modest PCMD approaches preferable to vision-heavy, dataset-tuned pipelines.

Scientific motivation: the temporality gap in procedural verification

At the core of procedure verification lies a temporal question: *what happened when, and did the relevant actions actually complete duly in the required order?* Procedural mistakes cannot

be reduced to appearance-based irregularities; they require long-range temporal reasoning over protocol structure. This gap becomes sharper when moving from online forecasting to post-completion verification: the model must reconcile the full trajectory, including order constraints, duration and termination, and potential recovery actions.

In linguistic semantics, temporal relations alone are not enough; verification also depends on the internal structure of events. Procedural steps often behave like Vendler-style accomplishments: they unfold over time and have a natural culmination (a *telos*) [95, 126]. The classic event-nucleus view decomposes an event into a preparatory process, a culmination, and a consequent state [84]. PCMD decisions hinge on distinguishing between an ongoing process and a completed step whose consequent state holds: the process may be visible while the culmination fails or the expected state never materializes.

As our diagnostics show in Chapter 3, current VLM-based pipelines often struggle with these fine-grained distinctions. Their failures reflect brittle temporal logic (confusing ongoing and completed actions, missing constraints implied by procedural structure) and unreliable decision behavior under common evaluation formats. Video–language benchmarks also frequently under-specify fine-grained temporal structure, which makes temporal failures hard to diagnose and can mask shallow strategies [17, 23, 71]. The broader video-LLM setting further amplifies these issues in long videos, where temporal dependencies and evidence aggregation are fragile and can interact with hallucination-like behavior [118]. These observations motivate the central thesis of this dissertation: reliable PCMD calls for structured and interpretable mechanisms over grounded step sequences, rather than black-box end-to-end VLM reasoning alone.

1.3 The Solution

To address these challenges, this thesis proposes a unified, **language-centered framework for PCMD**. The framework shifts procedural verification from purely visual end-to-end prediction to a structure-first approach: video executions are grounded into textual step sequences, and mismatch detection is performed primarily in text space with explicit temporal structure and calibrated decisions.

1.3.1 Pillar I: Diagnostics of temporal reasoning and evaluation reliability

The thesis first builds diagnostic evidence about where and why VLMs fail on procedural verification prerequisites. It studies temporal reasoning through temporal markers, grammat-

ical tense, aspect, and telicity, and analyzes reliability issues in video question answering (VideoQA) evaluation, including selection bias and "blind guessing" behavior. These diagnostics motivate shifting away from using VLMs as final arbiters of fine-grained procedural mismatches, towards structured verification over grounded step sequences.

1.3.2 Pillar II: Mistake-aware data and dataset assessment

The quality and quantity of annotated mistake-aware video data is a major bottleneck for scalable and generalizable PCMD. To address this limitation, the thesis introduces **PIE-V**, a semi-synthetic framework that uses psychology-informed planning to inject semantics-aware mistakes and plausible corrections into clean procedures. PIE-V is paired with a dataset assessment protocol that targets realism, coherence, and text–video grounding, supporting more reproducible evaluation and robustness testing.

1.3.3 Pillar III: Structure-first PCMD via a language-grounded HMM

Finally, the thesis presents a lightweight interpretable baseline for PCMD. It grounds video executions into step sequences and models them with a **Hidden Markov Model (HMM)**. Using a unified taxonomy based on edit operations, the method aligns intended and executed sequences and reports protocol-relevant mismatches as interpretable error types.

This thesis follows a single progression from diagnosis to construction and verification. Chapters 3 and 4 are not standalone excursions into VideoQA; they identify the reliability and temporal-binding failures that make end-to-end VLM verification brittle for PCMD. These diagnostic findings then motivate the two constructive parts of the thesis: Chapter 5 addresses the data bottleneck by building more realistic, instruction-grounded mistake traces, while Chapter 6 addresses the modeling bottleneck by shifting the verification decision into language-grounded step reasoning with explicit procedure structure.

1.4 Innovative Aspects

This thesis advances the state of the art by:

- **Diagnostic grounding of limitations:** a consolidated account of why end-to-end VLM reasoning is unreliable for procedural verification, explained in terms of temporal concepts, evaluation bias in VideoQA, and cross-modal misalignment patterns.

- **Scalable mistake-aware data methodology:** a rubric for assessing mistake datasets and a semi-synthetic pipeline (PIE-V) that injects semantically controlled mistakes and plausible corrections under a unified taxonomy.

- **Interpretable lightweight PCMD modeling:** an efficient, dataset-agnostic, structure-first baseline based on an HMM and edit-based diagnosis operating in text space, with clear error reports via edit operations.

1.5 Structure of the Thesis

The thesis is organized as follows:

- **Chapter 2** reviews background on procedural understanding, cognitive accounts of errors and recovery, VideoQA-based evaluation, and procedural mistake detection.

- **Chapter 3** and **Chapter 4** derive diagnostic requirements for PCMD by analyzing reliability failures in VideoQA-style evaluation and temporal-binding failures in current VLMs.

- **Chapter 5** responds to these requirements on the data side by introducing PIE-V, a framework for constructing and assessing mistake-aware procedural traces with realistic errors and recoveries.

- **Chapter 6** responds to the same requirements on the modeling side by introducing ChronoFix, a language-grounded, structure-first framework that combines HMM-based sequence verification with a representation study over raw step text, semantic roles, and action–object abstractions.

- **Chapter 7** concludes the thesis, summarizes how the three thesis pillars connect, and outlines future work.

1.6 Publications, presentations, posters and works under submission

The research carried out during this thesis has led to the following publications, manuscripts under review, and presentations.

Peer-Reviewed Conference and Workshop Papers

1. **Olga Loginova** and Frank Keller. *How to Correctly Make Mistakes: A Framework for Constructing and Benchmarking Mistake-Aware Egocentric Procedural Videos* [74] (**EgoVis** Workshop at **CVPR 2026**).

2. **Olga Loginova**, Oleksandr Bezrukov, Ravi Shekhar, and Alexey Kravets. *Addressing Blind Guessing: Calibration of Selection Bias in Multiple-Choice Question Answering by Video Language Models* [76] In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (**ACL 2025**).

3. **Olga Loginova** and Sofia Ortega Loguinova. *Deep Temporal Reasoning in Video Language Models: A Cross-Linguistic Evaluation of Action Duration and Completion through Perfect Times* [75] In Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (**ACL 2025**).

4. Gautier Dagan, **Olga Loginova**, and Anil Batra. *CAST: Cross-modal Alignment Similarity Test for Vision Language Models* [24] In Proceedings of the 31st International Conference on Computational Linguistics (**COLING 2025**).

5. **Olga Loginova** and Raffaella Bernardi. *The Inherence of Telicity: Unveiling Temporal Reasoning in Video Question Answering* [73] In Proceedings of the 9th Italian Conference on Computational Linguistics (**CLiC-it 2023**).

Manuscripts Under Review

6. **Olga Loginova**, Oleksandr Bezrukov, and Frank Keller. *Mistake Detection in Procedural Videos via Language-Grounded Step Reasoning*. Submitted, under review.

Selected Posters and Awards

- **BMVA Vision & Language Symposium 2025**. Poster: *PIE-V: Psychologically Inspired Error Injection in Instructional Videos*. **Olga Loginova**, Oleksandr Bezrukov, and Frank Keller. London, UK.

- **BMVA Symposium on Vision and Language 2024**. Poster: *Scaling VQA Datasets with Automated Temporal Annotation for Enhanced Temporal Reasoning*. **Olga Loginova**, Giovanni Bonetta, and Ravi Shekhar. London, UK.

- **Newton Gateway to Mathematics Workshop on Generative Models 2024**. Poster and Oral Presentation: *Revealing Hallucinations in Large Vision-Language Models through Self-Consistency*. **Olga Loginova**, Anil Batra, and Gautier Dagan. (Winner of the Hackathon on Generative Modeling, 2024, University of Edinburgh).

Chapter 2

State of the Art and Background

This chapter reviews the conceptual and technical foundations relevant to PCMD. It covers cognitive accounts of errors and corrections in routine procedures, procedural video understanding and step representations, how VLMs are commonly evaluated via VideoQA and where such evaluation can mislead, and the current landscape of mistake-aware datasets and mistake detection approaches. The chapter supports the three contribution lines of the thesis: diagnostics, data, and structured PCMD modeling; and motivates the design choices developed in Chapters 3–6.

2.1 Cognitive Grounding: Errors and Corrections in Routine Procedures

Procedural errors are a natural part of human behavior, especially in sequential, goal-directed activities. A classical distinction separates failures of execution from failures of planning. Slips and lapses describe situations where the intention is correct but execution or memory fails, while mistakes reflect incorrect plans or decisions [105, 106]. In routine procedures, these failures often manifest as omitted steps, unintended repetitions, wrong substitutions, or order disruptions. Such deviations are structured, not arbitrary: they correlate with limited attention, working memory constraints, interference, similarity-driven capture, and monitoring failures [15, 53, 88, 116].

A socio-technical perspective further frames errors as expected outcomes of complex systems: tightly coupled processes and high-risk settings make small deviations consequential [97]. This matters for procedural AI assistants because the target setting is not a controlled lab procedure but an environment where variability, interruptions, and partial observability are the norm.

2.1.1 Error types in sequential procedures

Reason's taxonomy separates *skill-based* failures (slips/lapses), *rule-based* mistakes, and *knowledge-based* mistakes [105, 106]. Norman further decomposes slips into categories tied to habit and similarity (capture errors), ambiguous descriptions, data-driven or associative intrusions, loss-of-activation, and mode errors [88]. These typologies are useful for PCMD because they foreground *where* in the procedure a failure tends to occur and *what kind of information* is typically missing (e.g., a monitoring check, a memory cue, or a task-state signal).

2.1.2 Post-completion errors: relevance and scope

A specific cognitive phenomenon is the post-completion error: omitting a final "cleanup" step after the primary goal appears achieved [15, 16]. This psychological feature provides a relevant error cues: humans can treat a procedure as complete even when a protocol-critical condition remains unsatisfied, because attention shifts away once the perceived goal is reached [16]. PCMD targets the computational problem that follows from this human tendency: a system must verify correctness *after* the execution appears finished, where errors can be subtle and only diagnosable from the full trajectory.

2.1.3 Instruction following, working memory, and cognitive load

Instruction following links perception, language, and executive control. Working memory capacity constrains how reliably people maintain step order, keep intermediate goals active, and monitor completion [15, 53, 139]. Longer and cognitively demanding procedures become more vulnerable to omissions and mis-ordering, especially under high intrinsic and extraneous load [2, 9, 29, 92]. Cognitive load theory formalizes this as an additive pressure on limited processing resources [2, 92]. Empirical and modeling work connects error likelihood to interruptions and load, with omission-prone behavior emerging when the system is close to capacity [15, 116].

For procedural video understanding, these accounts motivate two modeling commitments used throughout the thesis: (1) representing procedural correctness through explicit structure of steps, constraints, and temporal relations, and (2) treating corrections as first-class events rather than accompanying action.

2.1.4 Corrections, recovery behavior, and error management

Procedural executions often contain recovery. A person may commit an error and later correct it by redoing a step, inserting a missing one, or undoing an incorrect action. This implies that "incorrectness" is rarely a single isolated segment: it can propagate to later steps, and it can also be mitigated by subsequent repair actions. Error recovery in socio-technical systems highlights that correction behavior depends on the level at which the failure arises: skill-based errors are often easier to spot and correct, while rule- or knowledge-based mistakes can be detected later and are less likely to be repaired [105, 123].

These observations are directly relevant to PCMD. A model that flags a deviation must distinguish (1) an erroneous step, (2) its downstream consequences, and (3) later corrective actions. This thesis therefore treats corrections as part of the procedural semantics: they appear explicitly in the PIE-V generation framework (Chapter 5) and interact with sequence alignment and edit-based diagnosis in the PCMD model (Chapter 6).

2.2 Procedures as temporal–causal event structures

Procedures are not only ordered lists. They are temporal–causal structures in which steps constrain what can happen next, which intermediate states must hold, and what counts as completion. A temporal formalization describes relations between intervals (ordering, overlap, containment) [3]. Verification, however, also requires modeling the *internal structure* of events: whether an action has a natural endpoint and whether it reaches it.

2.2.1 Aspect, perfectivity, and telicity

In linguistic semantics, events differ in whether they are bounded and whether they culminate. Vendler's class of *accomplishments* captures a typical procedural step: it unfolds over time and has a necessary endpoint, such as *tighten the screw* or *boil the pasta* [95, 126]. The event-nucleus view decomposes such events into a preparatory process, a culmination, and a consequent state [84]. For verification, this decomposition is practical: the preparatory phase can look locally plausible while the culmination fails or the consequent state does not hold.

Perfectivity and telicity compactly express this distinction. Perfective morphology and telic predicates support completion inferences, while imperfective readings describe ongoing processes without commitment to the endpoint [95, 126]. Visual-world studies show that aspectual cues are integrated with perceptual evidence in real time: listeners shift attention toward depictions of completed outcomes when the verb form licenses completion, and

completion inferences can be delayed relative to basic lexical integration in child processing [13, 35, 36].

Completion entailments are also not uniform across languages. Perfective marking can be a strong completion signal in some systems (e.g., Slavic perfectives), whereas tense-only marking (e.g., English simple past) is a weaker and less reliable cue for culmination [83, 124]. This cross-linguistic asymmetry motivates treating completion as a *reasoning target*, not a superficial verb-form feature: PCMD hinges on whether a step merely *occurred* or was *completed* in a protocol-satisfying way.

2.2.2 Causality and culmination as verification signals

Procedures impose causal constraints: steps are ordered not only by time but by preconditions and effects. Crucially, procedural causality is often mediated by *culmination*: one event licenses the next once it reaches its endpoint, because the consequent state becomes the precondition for subsequent actions [84]. This aligns temporal verification with outcome verification: an execution can contain visually plausible motion while still violating the protocol if the required effect is not achieved.

Event knowledge models link actions to expected participants and outcomes, providing structured expectations about what should result from an event sequence [33]. For PCMD, this motivates outcome-sensitive checks: a step can be present as a process, yet invalid as an accomplishment if the telos is not reached or if the consequent state conflicts with what the protocol requires.

Taken together, aspectual boundedness (telicity/perfectivity) and temporal–causal constraints define what "completion" means in procedures. This view supplies the conceptual bridge between the cognitive grounding above and the VLM diagnostics in Chapter 3: procedural verification requires stable representations of completion, ordering, and effects, and it benefits from grounding these notions in both linguistic and perceptual signals [36, 83, 84].

2.3 Procedural video understanding: representations and supervision

Procedural video understanding covers long-horizon activities where a video must be decomposed into meaningful units and mapped to a task structure (steps, subgoals, state changes), not treated as a bag of short actions [82, 117, 156]. Temporal action segmentation formulations target this decomposition explicitly, combining fine-grained boundary localization with

higher-level activity regularities [27]. In parallel, procedure-aware representation learning couples video with narrations or instructions to encode step semantics beyond motion cues [155]. A reference point here is Ego-Exo4D [44], a large-scale egocentric–exocentric corpus of skilled activities with high-quality time-indexed language channels (e.g., narrations and expert commentary) and curated procedural annotations (e.g., keysteps). While not designed as a mistake benchmark, its expert commentary stream often functions as an execution critique, and the strong temporal alignment makes it a valuable substrate for grounding-intensive verification.

2.3.1 From actions to steps: protocols and step sequences

A protocol can be treated as an ordered step sequence with implicit preconditions and effects, where correctness depends on the global consistency of the realized trajectory. Datasets operationalize protocols differently: some provide full step text, some provide action labels, and some provide narrations that only partially cover the underlying protocol. For PCMD, step-level language is particularly informative because many deviations are structural (missing, repeated, swapped, substituted steps) and are most naturally diagnosed in sequence space rather than frame space [47, 96].

2.3.2 Step representations: from raw descriptions to semantic-role frames

Step representations vary along three axes: granularity (coarse vs. fine), linguistic variability (free-form vs. constrained), and semantic explicitness (what parts of the instruction are made machine-checkable). Raw human-written step text is expressive but underspecified for verification: the same intent can appear with different surface realizations, and crucial constraints (participants, instruments, result states) may be implicit.

A complementary representation treats each step as a predicate–argument (semantic-role) frame: a core action predicate plus role slots for the participants and relevant adjuncts (e.g., AGENT, OBJECT, INSTRUMENT, LOCATION, MANNER, DESTINATION, ORIGIN, DEGREE, RESULT and more). This is close in spirit to semantic role labeling (SRL) and to argument-structure accounts that connect verb meaning to participant roles [10, 42, 93]. For PCMD, role-structured steps provide two practical benefits:

- **Comparable mismatch types across domains.** Deviations can be described as role violations (wrong object, wrong tool, wrong target state), which stays stable when sur-

face step wording differs. This also makes it easier to align mistakes to *what exactly* is violated in the instruction rather than assigning a coarse error label [67].

- **Controllable edits for mistakes and corrections.** A role-slot view gives direct edit handles for synthetic perturbations (swap OBJECT, delete INSTRUMENT, substitute RESULT), which matches the needs of PIE-V (Chapter 5) and the representation study in the mistake detection module after the HMM pipeline (Chapter 6).

Finally, outcome-sensitive verification becomes easier to express when step meaning includes effects or result states: locally plausible execution dynamics can still produce the wrong effect, and this failure mode calls for tracking action consequences, not only motion patterns [46].

2.3.3 Supervision and grounding: aligning video with protocol structure

The usefulness of any step representation depends on whether the video execution can be grounded into a step sequence with reliable boundaries and identities. Weak alignment between segments and protocol language adds ambiguity: a model may recover an approximately correct step list but still lose the evidence needed for verifying completion, ordering constraints, and corrections. This motivates keeping the grounding interface explicit (video $\rightarrow$ step sequence) and evaluating it separately from downstream mistake diagnosis (Chapter 6).

2.3.4 Structure learning: graphs, constraints, and verification

Procedures can also be represented as structured objects (task graphs, state machines, constraint systems) rather than flat lists. Such representations support verification queries over executions and can express alternative valid paths or optional steps [60, 87, 110]. This thesis follows the same verification-first direction, but treats language-grounded step sequences (and their semantic-role structure) as the primary space where protocol constraints and deviations are stated, aligned, and diagnosed. This choice directly sets the stage for Chapter 4. By enforcing explicit step grounding, we compel the system to resolve temporal completion, ordering, and causal effects—precisely the capabilities that end-to-end VLMs lack, as evidenced by their unreliable behavior in diagnostic VideoQA settings.

This verification need becomes even sharper in the emerging vision-language-action (VLA) paradigm, where models must not only describe procedures but also act in the world. As VLA systems move toward executing long-horizon tasks, reliable post-hoc and in-the-loop verification becomes a safety-critical capability rather than a benchmark feature [154]. In

such embodied contexts, PCMD serves not merely as a retrospective debugging tool, but as a critical **safety monitor** that prevents execution failures from propagating into physical risks.

2.4 How VLMs are evaluated through VideoQA

VideoQA is a widely used interface for evaluating VLMs: a model answers natural-language questions about a video, and answers are scored either in open form or via multiple-choice selection [54, 153]. The format appears to probe reasoning about events, relations, and time, yet for procedural verification it can blur two distinct questions: whether the model can solve a temporal query, and whether the benchmark forces the model to use video evidence rather than shortcuts.

2.4.1 VideoQA benchmarks for temporality and procedural structure

Temporal and causal question types ("before/after", "during", "as") target event understanding and are often treated as proxies for procedural competence [51, 136]. In procedural settings, such questions approximate verification sub-skills: recognizing whether a step happened, whether it completed, how it relates temporally to another step, and whether the realized sequence is consistent with a protocol. Dedicated temporal benchmarks increase coverage of temporal relations and boundary cases that static-image reasoning cannot capture [17, 23, 71]. This thesis uses VideoQA primarily as a *diagnostic lens* (Chapter 3): it isolates temporal and procedural failure modes that later reappear in PCMD, while also exposing evaluation artifacts that can inflate apparent competence.

2.4.2 Multiple-choice VideoQA: convenience, selection bias, and "blind guessing"

Multiple-choice VideoQA (MC VideoQA) reduces the output space to a fixed option set, simplifying evaluation and comparison. However, the MC format amplifies dataset artifacts and model priors: correlations between answer options and question templates, positional effects, lexical regularities, and option-length patterns can become strong enough that accuracy no longer reflects video grounding. This phenomenon is closely related to *selection bias* in MCQA, a systematic preference for particular options or positions that persists even when evidence is weak or absent [66, 72, 98, 127, 152].

Selection bias has been extensively documented for text-only LLMs, including position and token biases and the use of prior-separation strategies for debiasing [152]. For vision-

language MCQA, the space of dedicated studies is smaller, but unimodal shortcuts have been shown to meaningfully affect measured performance: strong answers can be selected from language priors even when visual evidence is suppressed or uninformative [137, 146, 147, 158]. For video-language MCQA, this concern becomes sharper: videos increase computational cost and introduce additional degrees of freedom (temporal unfolding, camera motion, object persistence), while MC benchmarks may still allow high accuracy from question-only or weakly grounded heuristics.

From the perspective of procedural verification, this creates the failure mode of **blind guessing**: correct answers can be produced with limited reliance on the video stream, creating an illusion of temporal reasoning. Empirically, bias can be studied by comparing standard inference to *blind* zero-question settings and by controlled perturbations of options, questions, or video evidence [127, 147]. Debiasing approaches fall into several broad families that matter for interpreting VideoQA results: (i) prior modeling / separation (e.g., controlling for option priors via shuffling-based protocols) [152], (ii) benchmark augmentation via option re-ordering or expansion to dilute positional artifacts [72, 127], (iii) unimodal-bias isolation (e.g., suppressing visual input to estimate language priors) [146, 147], and (iv) confidence-based calibration that redistributes probabilities based on uncertainty estimates [98]. These ideas motivate the diagnostic protocol in Chapter 3, where MC VideoQA is treated as a biased measurement channel rather than a direct readout of video reasoning.

2.4.3 Consistency as a reliability diagnostic beyond accuracy

Accuracy on VideoQA benchmarks can mask instability: the same underlying video evidence can predict different answers when the input is perturbed in meaning-preserving ways. One common operationalization is self-consistency, tested via alternations such as paraphrasing, adding filler tokens, or injecting irrelevant statements while preserving the intended meaning of the query [32, 94, 143]. A complementary notion is logical consistency, which requires that repeated predictions remain coherent and non-contradictory across iterations and equivalent formulations [138, 148]. For procedural verification, however, stability must be assessed not only within the text channel but also across modalities. A model may remain internally consistent while drifting away from video evidence, producing stable yet weakly grounded answers. This motivates **cross-modal consistency** diagnostics, where consistency is evaluated through comparison-based judgments over paired examples rather than single-instance answering. CAST instantiates this idea by probing whether a model makes consistent similarity judgments when the semantic overlap between two visual inputs should support the same textual decision.

Comparison-style evaluation has also been used in vision–language benchmarks built around contrastive pairs, where visually similar inputs differ in a small number of key attributes or relations [30, 38, 150]. In the procedural domain, analogous comparison probes help separate genuine grounding from template-driven answering: consistency becomes informative precisely when the model's uncertainty would otherwise lead to unstable outputs.

Consistency is also closely tied to uncertainty and hallucination behavior: higher uncertainty can produce noisier and less consistent generations [19]. For this reason, consistency-based checks have been used as signals of misalignment and hallucination risk [65, 78, 86]. In trustworthiness-style diagnostics, inconsistency is frequently used as a practical proxy for hallucination: when grounding is weak, the same claim tends to break under equivalent queries or alternate evidence views [19, 78, 86]. In PCMD, this translates into a concrete failure mode: an ungrounded VLM may contradict itself when asked about the same deviation from different angles, motivating verification signals that are stable under re-querying and rephrasing.

Furthermore, such external inconsistency often correlates with the entropy of the model's internal states [19], which reinforces the need for the decision calibration mechanisms we propose in Chapter 6.

While such diagnostics do not guarantee factual correctness in the absence of ground truth, they provide a practical reliability lens for VideoQA-based evaluation and motivate the calibration and bias-aware analyses adopted later in this thesis (Chapter 3).

Chain-of-thought is not a temporal fix. Multimodal chain-of-thought prompting can improve explanation fluency, but it does not guarantee faithful temporal grounding: models may produce plausible reasoning traces while still relying on priors or incomplete evidence [132]. For procedural verification, this reinforces the value of explicit structure, where intermediate decisions (alignment, transitions, edits) have checkable semantics rather than free-form rationales.

2.4.4 Decision Calibration and Signal Detection Theory

Both VideoQA and PCMD can be viewed as decision problems under uncertainty: models trade off misses and false alarms when deciding whether evidence supports a hypothesis. Signal Detection Theory separates sensitivity from decision threshold and makes explicit the role of calibration [77]. This perspective is useful for procedural verification because it motivates thresholded decision-making and calibration on correct-only data, later used in the PCMD model (Chapter 6). It also motivates reliability-focused evaluation protocols in human judgment studies, which is relevant to dataset assessment and plausibility audits in PIE-V (Chap-

ter 5) [4, 104, 145].

However, many of these evaluation pathologies ultimately surface because temporal representations remain brittle: models may recognize relevant content without reliably binding it to order, overlap, and completion, which we discuss next.

2.5 Temporal reasoning for procedures

Temporal reasoning for procedures is not limited to ordering events along a timeline. For verification, it is crucial to represent (a) temporal relations between events and (b) the internal temporal structure of events, including completion and boundedness. Linguistically, this corresponds to the interaction of tense, aspect (including perfectivity), and telicity [84, 95, 126].

2.5.1 Temporal relations: ordering, overlap, and duration

Procedural steps impose constraints on what can happen next and what may overlap. Some actions are sequential by necessity; others can interleave without violating the protocol. Temporal relations therefore function as verification signals: a detected step may be present but occur at an invalid time relative to others, or it may occur without reaching its intended effect [3].

2.5.2 Completion as a verification cue

Verification depends on distinguishing "in progress" from "done" in a way that respects event structure. Completion judgments combine evidence about the preparatory process, the culmination, and the consequent state [84]. This motivates diagnostic probes that explicitly test completion- and culmination-sensitive reasoning in VLMs (Chapter 4) and motivates PCMD approaches that avoid relying on brittle implicit completion cues alone (Chapter 6).

2.6 Mistake-aware datasets and benchmarks

Mistake detection depends on datasets that expose models to protocol deviations and provide supervision beyond standard action recognition. Compared to mainstream procedural corpora (Table B.1), existing mistake-aware resources differ along three practical axes: scale (how many videos/tasks are available), protocol expressivity (whether steps are provided as natural language or only as normalized action labels), and mistake supervision (whether errors are labeled globally, localized in time, and whether they are binary or typed). This heterogeneity

shapes what can be learned, how models generalize across domains, and what kinds of failure modes can be evaluated [6, 67].

A spectrum of supervision. Existing benchmarks differ in how tightly they connect mistakes to the underlying protocol. At one end, mistakes are treated as *binary* sequence validity without localized diagnosis. For instance, SVIP/CSV labels whether an entire execution matches a reference protocol, emphasizing global structural deviations while leaving the source of failure implicit [101]. A more informative regime marks *where* errors occur by attaching binary correctness flags to action/step segments (e.g., "correct" vs. "mistake"), which supports temporal localization but still collapses diverse deviations into a single label [62, 131].

Richer datasets introduce *typed* mistakes that mix structural deviations (omission, repetition, swap, substitution) with executional failures (wrong object, wrong technique, timing/temperature issues), often through domain-specific taxonomies [47, 61, 96]. In these settings, the protocol is frequently present as step text, which enables linking an error not only to time, but also to the intended instruction.

Instruction grounding and explanation. For PCMD, mistake labels become substantially more actionable when they are tied to *what exactly* is violated in the instruction (action, object, instrument, manner, order constraint), rather than being a coarse tag attached to a segment. Some datasets therefore add natural-language explanations aligned to step descriptions (or implicitly referencing them). In this case models can learn mistake *semantics* rather than only anomaly signatures [47, 96]. Still, explanations are often free-form, unevenly grounded, and difficult to convert into structured supervision that is comparable across datasets. This motivates instruction-anchored formulations where mistakes are attributed to violated semantic roles and task constraints [67].

Corrections as first-class events. Procedural errors frequently co-occur with recovery: an incorrect step may be followed by undoing, redoing, or compensatory actions. If these are not explicitly represented, models can incorrectly penalize corrections as additional mistakes, or miss the fact that an early error was repaired later. A small subset of datasets includes explicit correction labels or correction-relevant annotation channels (e.g., detaching a wrong part [26, 111], conversational interventions [131]), which is particularly important for post-completion verification [61, 62, 131].

Not all supervision useful for PCMD comes from dedicated mistake datasets. Ego-Exo4D [44], for example, provides time-indexed expert commentary that mentions suboptimal

or incorrect execution decisions.

Benchmarking constraints. Three dataset-level constraints repeatedly surface as bottlenecks for PCMD:

- **scale and variance:** mistake instances are sparse and costly, and staged mistakes can introduce visual shortcuts and distribution artifacts [67, 96];
- **protocol text quality:** action-label protocols (verb–object tags) often underspecify constraints needed for semantic verification compared to step text [6];
- **text–video grounding:** weak alignment between steps and segments makes it hard to ground executions into step sequences, which is a prerequisite for language-grounded diagnosis; moreover, even when step/segment timestamps are available, fine-grained localization of *error onset and completion* is often missing beyond step-level boundaries [6, 47].

Table 2.1 summarizes representative mistake-aware procedural datasets in a supervision-oriented view. Datasets with only *video-level correctness* labels (e.g., SVIP/CSV [101]) support global verification but provide limited training signal for localizing mistakes or characterizing their type. Other datasets treat mistakes as *binary anomalies* at the segment level (e.g., ATA [41], BRIO-TA [85]), which is compatible with open-set detection objectives but typically does not explain what instruction constraint was violated. A third group provides *step- or segment-level mistake labels* and may expose correction behavior, either explicitly (through correction tags) or implicitly (through surrounding steps and narration/interaction traces) [26, 111, 131].

For PCMD, protocol representation is a central constraint. When the protocol is available only as action labels, mistake descriptions are limited to coarse surface forms (often verb–object phrases), which makes it harder to express instruction-sensitive deviations (wrong argument, wrong instrument, wrong manner). Datasets that include natural step text enable language-grounded verification and support richer mistake characterization, including protocol-aligned explanations [47, 61, 96]. Finally, scaling mistake supervision can rely on constructing mistake instances over existing egocentric corpora rather than collecting all errors organically; MATT [67] follows this direction and is therefore listed separately as a constructed benchmark.

These dataset properties motivate the assessment rubric and semi-synthetic enrichment pipeline introduced in Chapter 5, which explicitly targets realistic deviations and corrections and protocol-aligned, comparable supervision for PCMD.

Table 2.1: Mistake-aware procedural video datasets (compact supervision view). **#V**: videos; **#T**: tasks/scenarios; **#S**: step/action classes when reported. **Step ann.**: *Step* = natural step text, *Act* = action labels; *+ts* indicates timestamps. **Err. sup.**: *V* = video-level, *Seg* = segment-level, *Step* = step-level; *Bin* = binary, *Typed* = multi-class. **Expl.**: protocol-aligned natural-language explanations. **Corr.**: explicit correction labeling.

Dataset	#V	#T	#S	Domain	Step ann.	Err. sup.	Expl.	Corr.
SVIP/CSV [101]	70	14	106	chemical experiments	Act	V–Bin	×	×
ATA [41]	141	3	15	toy assembly	Act	Seg–Bin	×	×
BRIO-TA [85]	75	1	23	toy assembly	Act+ts	Seg–Bin	×	×
EPIC-Tent [50]	24	1	38	tent assembly	Act+ts	Seg/Step–Bin	×	×
Assembly101 [111]	362	101	202	toy assembly	Act+ts	Seg/Step–Bin	×	✓‡
IndustReal [107]	84	2	75	toy assembly	Step+ts	Step–Bin	×	×
HoloAssist [131]	350	20	414	AR-assisted manipulations	Step+ts	Seg–Bin	×	✓
CaptainCook4D [96]	384	24	352	cooking	Step+ts	Step–Typed	✓	×
EgoPER [61]	386	5	70	cooking	Step+ts	Step–Typed	×	✓
EgoOops [47]	40	5	46	various	Step+ts	Step–Typed	✓	×
MATT† [67]	–	×	–	various	Step	Step–Typed	✓	×

†MATT is constructed by augmenting existing egocentric corpora; counts depend on the chosen source subset.

‡Corrections are derived from undo/detach-style recovery actions [26].

2.7 Approaches to mistake detection

Procedural mistake detection methods vary most along two practical axes: **when** the decision is made (online/early vs. post-completion) and **what** is treated as the primary evidence for correctness (vision-first signals vs. explicit procedural structure). Across mistake-aware benchmarks such as EgoOops [47], CaptainCook4D [96], and EgoPER [61], this typically amounts to temporal modeling over recognized actions/steps and deciding when an observed stream departs from an expected procedural flow. A second-order difference is how "correctness" is represented: as a discriminative classifier over labeled error types, or as a normality model that flags deviations from learned correct behavior [6].

2.7.1 Online detection vs. post-completion verification

Online and early settings treat mistakes as anomalies in an unfolding action stream: given the prefix up to step $t-1$, the system decides whether the current observation is compatible with what should happen next, without using future context. A common pattern is to (1) recognize the current step/action, (2) maintain a belief over feasible next steps through a learned or induced task structure, and (3) flag low-probability transitions or out-of-distribution actions as

mistakes. PREGO [34] and TI-PREGO [99] instantiate this logic with task-graph-conditioned expectations and open-set deviation detection in egocentric assembly streams. Differentiable Task Graph Learning (DTG) [110] makes the task graph itself learnable from demonstrations and injects it at inference as a structured prior for online mistake detection. Related assembly-oriented formulations maintain symbolic or hybrid beliefs over feasible intermediate states and flag steps that violate the evolving procedure state [26]. AMNAR [49] extends the "what comes next" idea by modeling multiple plausible normal continuations and matching the ongoing action to the closest reconstructed normal representation, again under partial evidence. These online formulations align naturally with assistant-style settings where the system must monitor execution and potentially trigger interventions while the user is acting [131]. In all of these, the core constraint is the absence of look-ahead: recovery actions and delayed consequences can be difficult to interpret when only prefixes are available.

PCMD assumes the full execution is available and shifts the objective from anticipation to **verification against a protocol**. Full-sequence access enables explicit *sequence-level* reasoning (alignment to a reference procedure, edit-based diagnosis, aggregation of weak local evidence) and makes it possible to interpret recovery actions retrospectively (e.g., an early omission that is later repaired) [6, 123]. PCMD pipelines therefore often combine temporal localization of steps/segments, an explicit notion of expected structure (script, task graph, or step sequence), and a diagnostic layer that explains deviations.

2.7.2 Vision-first pipelines vs. structure-aware approaches

Vision-first pipelines learn mistake signatures directly from video features and dataset-specific supervision. This includes segment-level discriminative training for error categories (common in domain-tied taxonomies), as well as modality-driven normality modeling where deviations in embeddings, attention, or auxiliary signals are treated as mistakes [49, 79]. Such approaches can work well when error modes are well covered by the dataset labels and visually distinctive. Their main limitation in procedural settings is that a visually plausible action stream can still violate the protocol in *structural* ways (skip/swap/substitution), and these violations are often easier to express and diagnose in *procedure space* than in raw feature space. While end-to-end VLMs attempt to learn these structural constraints implicitly from data, they often fail to capture the rigorous temporal logic required for verification, as we demonstrate in Chapter 4.

Structure-aware approaches make the protocol explicit and treat mistake detection as incompatibility with a constrained procedure model. SVIP/CSV [101] formalizes verification at the sequence level by judging whether an execution conforms to a reference protocol. Task-

graph formulations [60] generalize this idea by representing procedures as graphs that can encode alternative valid paths and support error recognition beyond a single canonical sequence. Outcome- and state-centric modeling further emphasizes that certain errors are better diagnosed through *effects* (state changes, end states, object configurations) rather than motion dynamics alone: action-effect modeling flags mistakes when the predicted/observed effects do not match what the procedure entails [46], and state-change counterfactual objectives explicitly target what changed (and what plausibly could have changed) under procedure constraints [56]. In industrial settings, this idea can appear as explicit reference-state comparison, where deviations are localized by comparing observed states to correct reference states [62].

VLM-based pipelines sit between these families: they provide strong *video-to-text grounding* and can leverage scripts or step descriptions directly at inference, but their reliability depends on temporal reasoning, grounding stability, and decision calibration (Chapter 3). Some approaches use VLMs or LLMs to judge completed procedures and produce explanations conditioned on the script [114], while others combine textual protocols with segment-level classification in multimodal datasets [47]. These trends motivate a decomposition where the VLM is used primarily for grounding (video $\rightarrow$ step sequence), and verification/diagnosis is done with an explicit, calibrated procedure model.

Table 2.2 summarizes representative methods in a compact, supervision- and structure-oriented view. The placement is not meant as a taxonomy of all variants, but as a map of dominant modeling choices that later chapters reuse and recompose.

This thesis follows the structure-aware direction, but separates *grounding* from *verification*. Chapter 3 shows why temporal reasoning and decision reliability can break inside end-to-end VLM evaluation; Chapter 5 addresses data realism, comparability, and correction modeling; and Chapter 6 proposes a language-grounded, computationally modest PCMD baseline where the video is first grounded into a protocol-aligned step sequence and mistakes are diagnosed through an explicit probabilistic sequence model.

2.8 Summary of gaps and requirements for PCMD

Taken together, the background above highlights three requirements for progress in PCMD:

1. **Temporal–causal verification signals.** Procedural correctness is not reducible to "did an action occur": it depends on ordering and overlap constraints, on whether steps *culminate* (completion as accomplishment), and on whether the intended effects and consequent states hold. This creates a gap between surface temporal querying and verification-grade reasoning about completion, outcome, and causal preconditions.

Table 2.2: Representative procedural mistake detection methods (compact view). **Set.**: setting (On/Post). **Str.**: injected structure. **Sig.**: core signal. **Sup.**: supervision type. **Bench.**: evaluation benchmarks (abbrev.).

Method	Set.	Str.	Sig.	Sup.	Bench.
PREGO [34]	On	TG	A	B	A101, ET
TI-PREGO [99]	On	TG	A/LLM	B	A101, ET
DTG [110]	On	TG	A	B	A101, ET
Every Mistake Counts [26]	On	ST	Cns	B/T	A101
AMNAR [49]	On	–	A	B	A101
SVIP/CSV [101]	Post	RP	Ver	B	CSV
StateDiffNet [62]	Post	ST	SD	B	Ind (asm.)
ED-VLP (EgoPER) [61]	Post	Prt	P	T	EPR
Weakly-sup. unseen errors [41]	Post	–	A	W	ATA
Gaze-based normality [79]	Post	–	Gz	C	ego (skill)
Action-effect modeling [46]	Post	Eff	E	B/T	proc. err.
Generalized Task Graph [60]	Post	TG	Ver	B/T	proc. vid.
Script-conditioned VLM [114]	Post	Scr	L	B/T	text+vid

Set.: On=online/early, Post=post-completion. **Str.**: TG=task graph, ST=state model/reference, RP=reference protocol, Prt=step prototypes, Eff=effects/state-change, Scr=script/step text, –=none. **Sig.**: A=next-step anomaly/normality, Ver=sequence/protocol verification, Cns=constraint/state consistency, SD=state difference, P=prototype deviation, E=effect mismatch, Gz=gaze deviation, L=language (script-conditioned judgement/explanations), A/LLM=anomaly with LLM/VLM prompting. **Sup.**: C=correct-only, B=binary, T=typed, W=weak/unsupervised. **Bench.**: A101=Assembly101, ET=EPIC-Tent, EPR=EgoPER, ATA=ATA, CSV=SVIP/CSV, Ind (asm.)=industrial assembly setups.

2. **Mistake data with realism and comparability.** Existing mistake-aware datasets differ in scale, protocol expressivity, and supervision regimes (video-level vs. segment/step-level; binary vs. typed), and only a subset explicitly represents corrections or provides protocol-aligned explanations. Progress requires supervision that is comparable across domains, grounded to what is violated in the instruction (including semantic-role-level mismatches), and paired with measurable text–video grounding quality.

3. **Interpretable, efficient models in procedure space.** Vision-heavy, dataset-tuned pipelines can be expensive and hard to diagnose when they fail, while MC VideoQA-style evaluation can inflate apparent competence through bias and blind guessing. PCMD therefore benefits from models that separate grounding from verification, operate explicitly on protocol-aligned step sequences, and use calibrated decision rules to diagnose deviations (omissions, insertions, substitutions, reorderings, and recovery) in

a transparent way.

These requirements shape the contributions developed in Chapters 3–6: diagnostic evaluation of temporal and decision reliability, data enrichment and assessment for realistic mistakes and corrections, and a structure-first, language-grounded PCMD model that performs verification and diagnosis in text space.

Chapter 3

Diagnostics: Reliability Pitfalls in VideoQA-Style Evaluation for PCMD

This chapter studies a practical obstacle to PCMD: even when a VLM can recognize relevant content, standard evaluation interfaces can make it appear reliable while it is in fact making decisions driven by priors and artifacts. The central claim is that PCMD is a verification problem: a thresholded decision about correctness under temporal–causal constraints, and therefore it is particularly sensitive to (1) multiple-choice selection bias when assessed through Accuracy, (2) hallucination-prone inconsistencies, and (3) weak cross-modal grounding.

Provenance. Parts of Sections 3.2–3.2.5 are adapted from our ACL 2025 work on multiple-choice selection bias in VideoQA. Sections 3.3–3.3.4 draw on our CAST study on cross-modal inconsistency.

3.1 What PCMD requires from reliability and decision-making

PCMD systems ultimately make a decision of the form *"is this execution correct with respect to a protocol?"* This decision is often thresholded: a system must flag deviations (that represent mistakes) while controlling false alarms. This amplifies two reliability requirements.

Requirement 1: evidence must be separable from priors. If a model can guess correctly from question/option regularities, then accuracy does not imply video grounding, and any downstream threshold inherits that unreliability.

Requirement 2: decisions should be stable under equivalent queries. If a model contradicts itself under rephrasings or alternate views of the same evidence, it is likely operating with weak grounding. In trustworthiness-style analysis, inconsistency is often a practical proxy for hallucination [19, 78, 86].

These requirements motivate the diagnostics below and inform the later design choice to separate *grounding* (video $\rightarrow$ step sequence) from *verification/diagnosis* (sequence-level decision in an explicit procedure model).

3.2 Reliability in multiple-choice VideoQA: selection bias and "blind guessing"

Section 2.4.2 argued that multiple-choice evaluation can decouple measured accuracy from actual video grounding. Here we move from motivation to diagnosis: we treat MC VideoQA as a **decision channel** whose output can be systematically distorted by option priors. Concretely, we ask a simple reliability question: *if the task is made ill-defined by removing one essential component (video, question, or option content), does the model collapse to near-uniform guessing?* Empirically it does not. Instead, many VLMs retain stable preferences for option positions, making confident predictions even when evidence is missing. We refer to this as **blind guessing**.

This matters for PCMD because verification is typically *thresholded*: any persistent option prior acts like an uncalibrated confidence baseline, contaminating downstream decisions unless explicitly measured and removed.

3.2.1 Problem setup and why it matters for PCMD

Consider an MC VideoQA task composed of three independent channels: the video context V, the question text Q, and the answer options O. A task is *well-defined* only if the combination (V, Q, O) points to a unique correct option. If any component is removed (e.g., $V = \emptyset$), the task becomes ill-defined, and an ideal unbiased reasoning agent should produce a uniform probability distribution over the options, reflecting maximum uncertainty.

Empirically, modern VLMs do not behave this way. They exhibit systematic deviations from uniformity—stable preferences for specific option positions (e.g., Option C) or lexical features—even when the video or question is missing. Formally, we investigate the bias $Bias_M$ by observing the model's behavior on decomposed variants $A_{v=0}$ (no video), $A_{q=0}$ (no question), and $A_{o=0}$ (no option text). For PCMD, this is critical because verification is

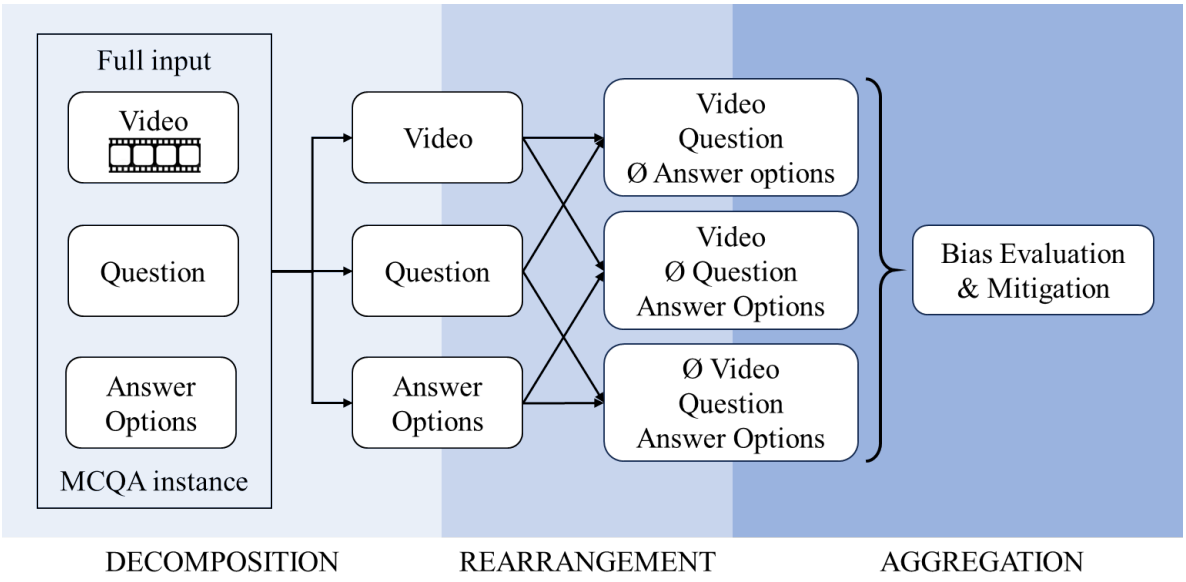


Figure 3.1: Decomposition idea: removing a key component makes the MC task ill-defined; non-uniform predictions under the ill-defined task reveal selection bias.

typically thresholded. A persistent option prior acts as an uncalibrated "confidence baseline," contaminating downstream decisions (such as "Mistake" vs "Correct") unless explicitly measured and removed.

3.2.2 Decomposition diagnostics: making bias visible

We diagnose selection bias by constructing ill-defined variants of each benchmark item via three "attacks" that remove one key component:

- $A_{v=0}$: the video is replaced by zero frames (video removed),
- $A_{q=0}$: the question is emptied (question removed),
- $A_{o=0}$: option texts are replaced by option IDs only (options-content removed).

As illustrated in Figure 3.1, we systematically isolate these components to measure the model's prior distribution when the task is structurally broken.

In a fair model, predicted option probabilities under these attacks should be close to uniform. Instead, the attacks expose strong option preferences. We quantify this using dispersion-style bias metrics (e.g., option-wise standard deviations of Recall and F1; probability-distribution dispersion via Jensen–Shannon distance).

For transparency, we provide concrete examples of all evaluation perturbations in Appendix A.1.3 (Table A.5).

3.2.3 Calibration via option-prior estimation (BOLD)

We estimate an option prior from ill-defined variants and remove it from the model posterior. Let $P_o(\cdot \mid T)$ be the model distribution over option IDs for the original task T. We estimate a global prior $\tilde{P}_p$ by averaging predictions on attacked variants:

$$\tilde{P}_p(d) = \frac{1}{K} \sum_{T \in \mathcal{D}_k} \text{softmax} \left(\sum_{A \in \{A_{v=0}, A_{q=0}, A_{o=0}\}} s_A(d \mid T) \right), \tag{3.1}$$

where $s_A(d \mid T)$ are model scores for option d under attack A. We then debias the original prediction by subtracting the log-prior:

$$P_d(d \mid T) \propto \exp\big(\log P_o(d \mid T) - \log \tilde{P}_p(d)\big). \tag{3.2}$$

Weighted_BOLD learns a weighted combination of attacked priors (Appendix A.1).

3.2.4 Results: bias drops, accuracy is more interpretable

Across multiple benchmarks, STAR [133], NExT-QA [134, 135], Video-MME [37], Perception Test [100], BOLD-style calibration reduces option-wise dispersion and often improves accuracy. More importantly, the calibration exposes when performance is inflated by priors: large improvements in bias metrics under ill-defined tasks indicate that pre-calibration accuracy partly reflected selection artifacts.

Full per-benchmark results (accuracy and bias dispersion metrics) for BOLD and Weighted_BOLD under different estimation budgets are reported in Appendix A.1, Tables A.1–A.4. Exact decomposition prompt formats and experimental setting variants are listed in Appendix A.1 (see Table A.5 and Sec. A.1.1).

3.2.5 Implications for PCMD: thresholding needs calibrated signals

The key takeaway is not merely "VideoQA is biased", but that MC bias directly contaminates thresholded verification. In PCMD, a decision rule should be built on signals with a clear evidential interpretation, such as sequence likelihood under a protocol model, edit structure and consistency of step grounding, rather than raw MC option selection.

3.3 Cross-modal inconsistency as a proxy for hallucination: CAST

Selection bias is one reliability pitfall; another is that a model can remain fluent and locally plausible while drifting away from the visual evidence. A practical way to expose weak grounding is to test whether the model is *consistent* when asked the same underlying question through meaning-preserving perturbations or through paired comparisons.

3.3.1 CAST setup: comparison-based consistency

CAST (Cross-modal Asymmetric Similarity Test) is a diagnostic that asks whether a model makes stable similarity judgments when only a small set of semantic attributes is changed between two visual inputs. The benchmark is built from controlled pairs: two short images (or videos) are designed to be semantically close, but differ in a key attribute; the model must answer a comparison query where the correct decision depends on grounding, not priors.

Figure 3.2 illustrates a full CAST constituent end-to-end. Given a paired input of visually and textually close items, the model first generates short similarity statements, and we then re-evaluate those statements under three evidence conditions (image-only, text-only, and image+text). Disagreements between the model's own generation and its subsequent evaluations (marked by red crosses) operationalize cross-modal inconsistency as a grounding diagnostic. Note that VLMs may produce hallucinations, as the CAST method checks for consistency rather than correctness.

CAST operationalizes grounding reliability as *consistency under controlled pairwise perturbations*: given a minimally-changed pair (x, x') and a comparison query q, a grounded model should preserve its decision whenever the semantic relation implied by q is unchanged. We measure consistency via flip rates across paraphrases/views and accuracy on the targeted attribute.

We introduce CAST (shown in Figure 3.3) as a fully automated two-step approach to evaluate multi-modal self-consistency in VLMs. It leverages similarities between two scenes to assess a model's ability to evaluate its own outputs. In our case, a scene is an image paired with its high-quality description from the DOCCI Dataset [90].

Unlike difference captioning, emphasizing similarities forces the model to perform a comprehensive scene sweep. For instance, if tasked with finding differences between two images, the model might only attend to one image or highlight minor details like color changes. Emphasizing similarities encourages a deeper evaluation of each input.

The first step of CAST is to prompt the VLM to generate a number of statements about the

A **close-up shot** of the right side of the face of a **cream-colored labradoodle** puppy looking to its right at the camera and its front paws on the back part of the seat of a wooden chair. A **grey tabby cat** is to the right of the dog facing the background and looking to its left at the dog on the brown chair, peering under the back rest of the chair…

A **close-up view** of a **cream-colored labradoodle** looking through the wired cable railing of a stair set as a **black and white cat** that is staring back, resting on the first step…

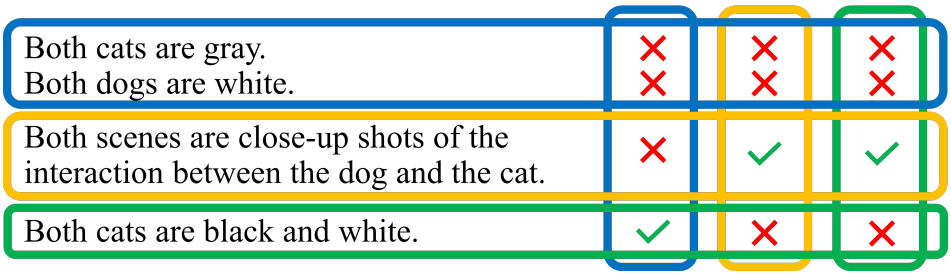


Figure 3.2: CAST example: paired inputs differ in a targeted attribute; comparison questions probe whether the model's judgment is stable under controlled semantic overlap. Horizontal blocks show generated statements, while vertical blocks are evaluations for each modality: image-only, text-only, and image+text. Red crosses indicate where each model disagrees with its own generation during the evaluation step. Similarity topics are highlighted **in bold**.

similarities between two input scenes S_A and S_B. Since we generate a list of similarities using the VLM, each subsequent similarity statement is conditioned on all previously generated ones. We can view the generation of a given similarity as follows:

$$sim_0 = VLM(S_A, S_B, P^{gen}) \tag{3.3}$$

$$sim_i = VLM(sim_{i-1}, ..., sim_0; S_A, S_B, P^{gen}), \tag{3.4}$$

where P^{gen} are the instructions. Similarity statements are generated for different modalities: scenes can be represented as two images (S^{img}), two text descriptions (S^{txt}), or two images combined with the corresponding descriptions ($S^{img+txt}$).

We obtain the similarity statements conditioned on a pair of scenes for each modality stream. We restrict the input pairs to the same modality and generate all statements using greedy sampling ($t = 0$).

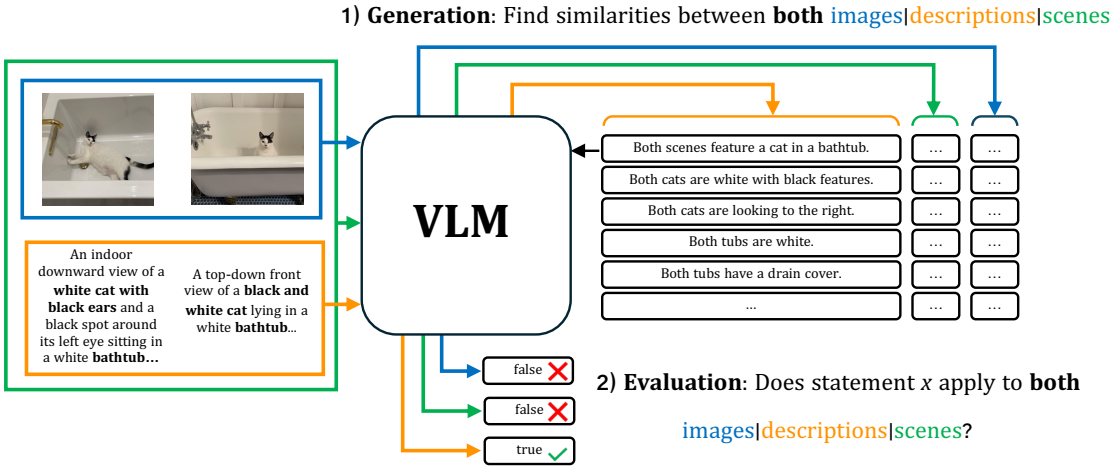


Figure 3.3: CAST is two-fold. In the first step, we ask the model to *generate* a set of similarity statements conditioned on different modality input types (image-only, text-only, both). In the second step, the model *validates* the truthfulness of the generated statements with respect to each modality. This allows us to measure whether the VLM is self-consistent within a modality and across different modalities.

3.3.2 Evaluating Similarities

The second step of our approach is to evaluate each similarity statement and test whether a model remains consistent under different modalities. Since we focus on self-consistency, we use the same model for both generation and evaluation. The evaluation step can be represented as follows:

$$s = VLM(S_A, S_B, P^{eval}), \tag{3.5}$$

where s is 1 if the model confirms that the statement is true and 0 otherwise. We filter out the generations that cannot be parsed. Prompt templates are given in Appendix A.2 (Section A.2.2). To mitigate bias towards a certain prompt or phrasing [98, 109], we use three different evaluation prompts. Thus, apart from the conventional Yes/No questions, we ask the model whether the statement applies to one or both scenes and whether the statement is true or false. To quantify self-consistency, we report the average s over all the evaluated pairs and prompts for each modality permutation (both generated and evaluated with).

3.3.3 Findings: inconsistency correlates with weak grounding

We tested the following open-source and closed-source VLMs for self-consistency, each with distinct vision encoders, language models, and training dataset: **Bunny 1.1** [48], **LLaVA** [70]

in three configurations: LLaVA 1.5 (Vicuna [21]), LLaVA 1.6 (Llama [122]), and LLaVA 1.6 (Mistral [52])[1], **InternVL2** [20],**MiniCPM V2** [140],**Phi 3.5 Vision** [1],**GPT4o-mini**[91].

Across tested models, CAST reveals systematic inconsistency: models can answer correctly on one phrasing or one view, but flip under equivalent comparisons, suggesting that the decision is not anchored in the visual evidence. This aligns with trustworthiness work in LLMs: inconsistency is often correlated with hallucination and overconfidence [19, 78, 86].

The inconsistency implies that traditional accuracy on VideoQA benchmarks is partly illusory. For PCMD, this necessitates the structured approach defined in Chapter 6, which relies on edit-distance alignment rather than black-box QA.

3.3.4 Implications for PCMD: verification should be stable under re-querying

For PCMD, CAST-style inconsistency translates into a concrete failure mode: an ungrounded VLM may contradict itself when asked about the same spatial deviation or when queried at different time scales (segment-level vs step-level). This motivates using structured verification signals, such as alignment, transition constraints, and edit costs, that remain stable under rephrasing and can be audited.

3.4 Summary: reliability constraints carried forward

The diagnostics in this chapter motivate three reliability constraints that later chapters adopt.

Constraint 1: MC accuracy is not evidence without debiasing. Decomposition exposes strong option priors; calibration makes downstream thresholding more meaningful.

Constraint 2: inconsistency is a red flag for weak grounding. Comparison-based diagnostics (CAST) show that fluent outputs can be unstable when evidence is weak, connecting inconsistency to hallucination risk.

Constraint 3: PCMD should use calibration-compatible signals. Verification is best performed in a space where intermediate decisions are checkable (step alignment, likelihood under a protocol model, explicit edit structure), rather than in opaque end-to-end option selection.

[1] Additionally, we evaluated LLaVA 1.5 RLAIF [142], a version of LLaVA 1.5 aligned through AI feedback.

On the data side, this means that mistake-aware supervision should be grounded in explicit procedural structure rather than isolated anomaly labels; on the modeling side, it means that verification should operate on protocol-aligned step sequences rather than on opaque end-to-end answers.

Taken together, these diagnostics show that a VLM answer is too unstable to serve as the final PCMD decision. They motivate the next part of the thesis: first a closer analysis of temporal reasoning failures in Chapter 4, then a shift to controlled data design improving training and evaluation of mistake-detecting models in Chapter 5 and explicit sequence-level verification for PCMD in Chapter 6.

Chapter 4

Diagnostics: The Temporality Gap in VLMs

This chapter targets a second prerequisite for Post-Completion Mistake Detection (PCMD): robust temporal and causal binding. VLMs often recognize *what* actions occur, yet fail to represent *when* they complete, *how* they relate (before/after/overlap), and *which consequent states* they license in a procedure. This gap matters for PCMD because many procedural mistakes are inherently temporal: misordering, premature transitions, missing culminations, and state-inconsistent continuations.

Provenance. Sections 4.2–4.2.4 adapt our CLiC-it 2023 paper on telicity biases in temporal VideoQA. Sections 4.3–4.3.4 are based on our BMVA 2024 Vision & Language Symposium poster on scaling temporal VideoQA via automated temporal annotation and template generation. Sections 4.4–4.4.8 adapt our ACL 2025 work on PERFECT TIMES about cross-linguistic perfectivity and temporal binding.

4.1 Conceptual framework: procedures as causal event chains

A procedure is not merely a list of action names. Procedural steps are constrained by preconditions and effects; many steps are accomplishments whose successful completion licenses subsequent actions. This makes **completion** a verification signal rather than an optional detail.

Event structure and completion. Vendler's event classes distinguish *accomplishments* (durative actions with an inherent endpoint) from *activities* (durative actions without a necessary endpoint) [126]. Moens and Steedman's event-nucleus decomposition separates a preparatory process, a culmination, and a consequent state [84]. In PCMD terms, a step may be visible as a process yet still be incorrect if its culmination fails or the consequent state does not hold.

Telicity vs perfectivity (why language helps diagnose video understanding). Telicity is primarily a *lexical/semantic* property: some predicates encode an endpoint internally (e.g., *assemble*, *close*). Perfectivity is primarily a *grammatical aspectual* packaging of event boundaries (e.g., perfective vs imperfective viewpoints). Procedural verification depends on both: lexical endpoints define what counts as completion, while aspectual cues signal whether an event is presented as completed, ongoing, habitual, etc. Multilingual diagnostics are therefore a useful stress test: if a model cannot use explicit completion cues in language, it is unlikely to infer completion reliably from video alone.

Across the three studies below, we probe whether models: (1) separate ongoing vs completed readings of the same event, (2) bind actions to the temporal moment required by a question (not merely "see" the action), and (3) handle boundary-sensitive relations (before, after, when or while; overlap and succession) beyond a narrow set of templates.

4.2 Study I: Telicity bias in temporal VideoQA

This study (CLiC-it 2023, *The Inherence of Telicity*) asks whether a strong VideoQA model exhibits systematic completion-like preferences when the question targets ongoing events. The core hypothesis is that VLMs over-associate procedural events with endpoints, implicitly treating "action observed" as evidence of completion.

Previous research explored telicity for textual transformer-based [125] models, showing that they can classify activities based on duration and telicity with an accuracy surpassing 80% [81]. This indicates transformer's ability to capture temporal reasoning through aspect classification. We extend this line of research to VLMs, where video content becomes the ultimate confirmation of telicity and therefore check its relevance for answering temporal questions related to simultaneous and consecutive activities.

4.2.1 Setup: telicity-annotated NExT-QA temporal subset

We analyze a temporal subset of NExT-QA and annotate events in the main clause and dependent clause of questions with telicity labels. This enables a targeted error analysis: when

questions are phrased to query *ongoing* events, do models still prefer telic (endpoint-leaning) interpretations?

We consider the aspects of question's both main and dependent clauses.

Q: what is the girl doing with her hands (A)
as the man moved his head sideways at the end of the video (T)

a0: smiles (A)	a1: jamming (A)	a2: pick up something (T)	a3: clap (A)	**a4: pick up toy (T)**

Figure 4.1: Example of a temporal question and answer options in NExT-QA augmented with our annotation for telic (T) and atelic (A) actions.

All question activities are divided into two groups: activities of the main clause (MCA) and activities of the dependent clause (DCA). Both question groups, as well as the target and predicted answers, were annotated independently with the following labels of the internal temporal structure:

- **T (telic)** for activities implying an endpoint (e. g., "what happened", "pick up camera", "after the door opens"),
- **A (atelic)** for enduring processes (e. g., "how is the person in black positioned", "smiles", "while watching"), and
- **U (undefined)** for activities lacking clear telicity and duration (e. g., "what does the dog do", "do the same", "to man's action to him").

Additionally, an **I (irrelevant)** marker was assigned to answers unrelated to aspectuality, such as "astonished" or "nothing". This marker appears among the target answers too in response to questions like "how did the boy react to..." or "what does the person do while...".

4.2.2 Model and protocol

We evaluate SeViLA [141] in a zero-shot setting on the annotated test set. Besides overall answer accuracy, we compute a reduced test subset (RTS) that allows mapping answer options to telic/atelic classes, making telicity-classification style metrics.

The example of an annotated question-answer pair from NExT-QA along with the SeViLA answer is given in Figure 4.1

The obtained results revealed the overall accuracy of 63.18% and the T-type question accuracy of 60.18%. On RTS, we obtained 58.1% of matching predicted and target answers. We further calculated the telicity precision, recall, F1 score and accuracy on the annotated RTS.

4.2.3 Results: telic-prone substitutions

Our analysis shows an asymmetric substitution pattern: telic answers substitute for atelic targets more often than the reverse. The model exhibits a clear inclination towards selecting telic values instead of the target atelic ones: in 26,12% of the target atelic answers it chooses the telic ones, while there are only 14.45% of the opposite cases. This is consistent with an endpoint-driven bias: even when questions target an ongoing interval, model decisions drift toward "completed" readings.

The full results are in Table A.7 (Appendix A.3).

The telicity cues may have their origins in the question's both MCA and DCA. As much as in the question about the next action SeViLA disregards the DCA's telic action, it also struggles to correspond with the atelic activities of the MCA in the answer for the ongoing action.

4.2.4 Implications for PCMD

For PCMD, telicity bias translates into a concrete verification risk: a step may be treated as "done" once its process is visible, even if the culmination and consequent state are missing. This motivates completion-aware representations and state checks in later chapters: PCMD should not reduce correctness to mere action recognition.

4.3 Study II: Scaling temporal VideoQA diagnostics with automated annotation

A practical obstacle to diagnosing temporal competence is that many existing temporal and causal VideoQA benchmarks probe a narrow set of constructions (often simple before/after or immediate "as soon as" triggers). This limited linguistic and relational coverage can mask brittleness: models may appear strong while failing on overlap, containment, boundary-touching cases, and more realistic paraphrases. We address the need for a more comprehensive coverage-first approach to temporal reasoning diagnostics with our scaling framework.

4.3.1 Construction: from timestamped actions to broad relation coverage

The framework takes as input any video corpus with time-stamped event annotations (such as procedural datasets) and generates temporal VideoQA instances by composing:

- **Event-time primitives:** pairs of actions with intervals or boundaries derived from timestamps;
- **Relation templates:** a broadened inventory of temporal relations, including Allen-style interval relations (overlap, meets, during, etc.) [3];
- **Distractors (for MCQA):** options that preserve action content but violate the required temporal relation;
- **Paraphrase augmentation:** LLM-based rephrasing to increase linguistic diversity while keeping semantics fixed.

The key idea (shown in Figure 4.2) is *scaling by quality*: instead of merely adding more samples of the same easy relations, we systematically cover a wider relation space and then test robustness under paraphrasing.

4.3.2 Why paraphrasing is not a cosmetic change

Template-generated questions are controllable but linguistically "artificial". To approach human-like dialogue while preserving the targeted temporal relation, we paraphrase each template question with an LLM: we generate multiple rephrasings and then sample one variant, providing diverse natural formulations.

This also produces a diagnostic lever: if model ranking flips between template and paraphrase variants, the benchmark is measuring surface-form sensitivity rather than stable temporal binding.

4.3.3 Results: low accuracy and robust sensitivity to form

Empirically, both SeViLA and IDEFICS3 [58] obtain low accuracy on these temporal variants, indicating difficulty with event-sequence reasoning even under MCQA evaluation. At the same time, both models show relatively stable behavior under paraphrasing in the settings we tested. This suggests that the main bottleneck is not only linguistic variation but the underlying temporal representation.

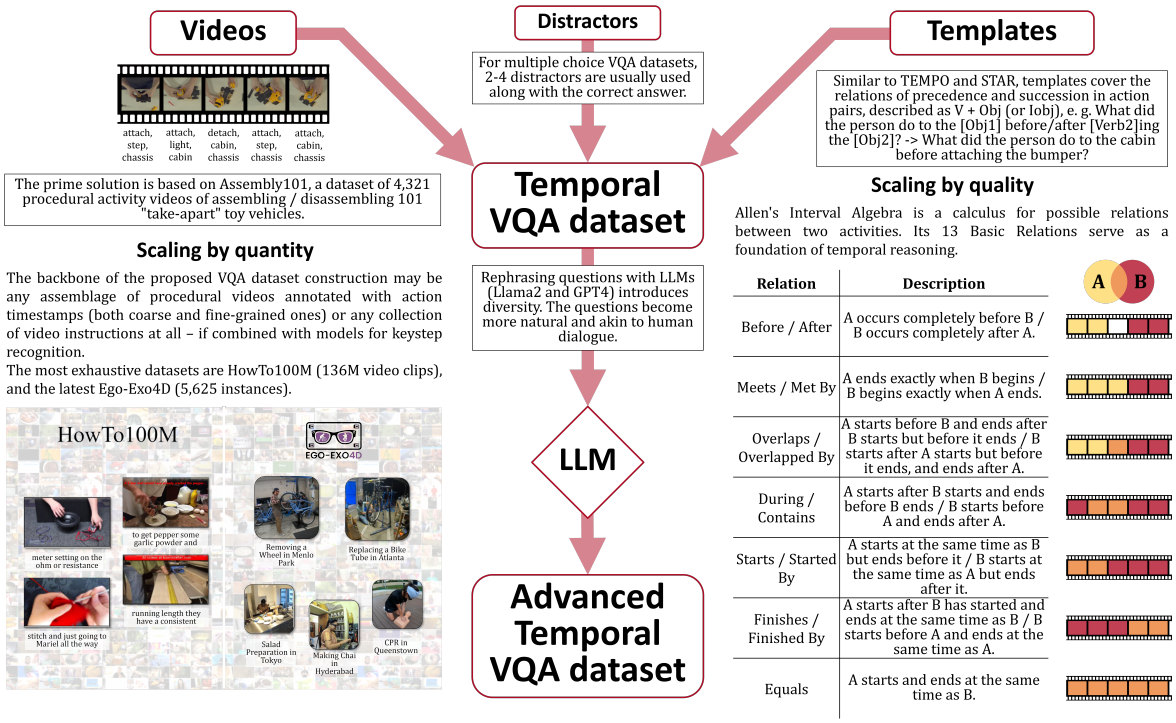


Figure 4.2: Scaling procedural VideoQA datasets with automated temporal annotation scheme.

4.3.4 Implications for PCMD

Scaling diagnostics supports a thesis-level point: evaluation must broaden relation coverage and control linguistic form. PCMD-grade verification should be tested under overlap/containment/boundary cases and under meaning-preserving paraphrases, because brittle temporal representations can be masked by narrow templates or dataset priors.

This study also motivates the next one: once relation coverage is addressed, we can probe whether *explicit aspectual marking* (perfective/imperfective) helps models bind video evidence to temporal logic.

4.4 Study III: Perfectivity and cross-linguistic deep temporal reasoning in VLMs

This study on PERFECT TIMES asks whether explicit linguistic marking of completion and duration helps VLMs bind video evidence to temporal relations. The central hypothesis is that many VideoQA successes reflect *action recognition without reliable temporal–causal binding*: models may identify *what* happens in a clip, yet fail to determine *when* an event

Table 4.1: Poster diagnostic: accuracy under template-based temporal VideoQA and paraphrased variants. **Templates + LLama2 (STAR)** means: STAR is template-built; we paraphrase STAR questions with LLama2. **LLama2 (NExT-QA)** means: we paraphrase NExT-QA questions with LLama2.

Benchmark variant	SeViLA acc. (%)	IDEFICS acc. (%)
Templates (Assembly101-QA)	34.6	61.6
Templates + LLama2 (Assembly101-QA)	34.5	61.5
Templates + LLama2 (STAR)	43.5	35.7
LLama2 (NExT-QA)	51.2	37.2

culminates, and *which temporal relation* holds between two events (precedence, succession or overlap).

4.4.1 Motivation: tense, perfectivity, and observation as language-specific trade-offs

Reasoning about temporal and causal relations between actions typically relies on three sources of information: **temporal anchoring** (tense and explicit temporal markers such as *before*, *while*, *after*); **aspectual viewpoint**, in particular **perfectivity**, which encodes whether an event is presented as completed/culminated vs. ongoing; and **observation**, i.e., perceptual evidence that confirms whether a state change actually occurred.

Crucially, languages allocate this explanatory burden differently. In English, temporal anchoring and perfectivity marking are informative but often leave room for contextual resolution, keeping observation as an equal partner in deciding whether an event has actually completed. Italian tends to encode temporal sequencing and event status more explicitly through a richer inventory of tense/aspect/mood and advanced sequence-of-tense constraints (*concordanza dei tempi*), so more of the ordering and completion information can be recovered from the utterance itself. Russian shifts weight toward perfectivity expressed through lexical choices and aspectual verb pairs, which can make completion contrasts especially salient in the surface form. Japanese, by contrast, can leave completion-like readings more dependent on context, increasing reliance on observation for disambiguation.

Figure 4.3 schematically summarizes these differences and motivates multilingual temporal VideoQA as a principled stress test: if a model genuinely binds to the video, performance should be broadly stable across synchronized question families, whereas large language-specific deviations suggest that the model is extracting temporality from the text tokens rather than from the underlying visual evidence.

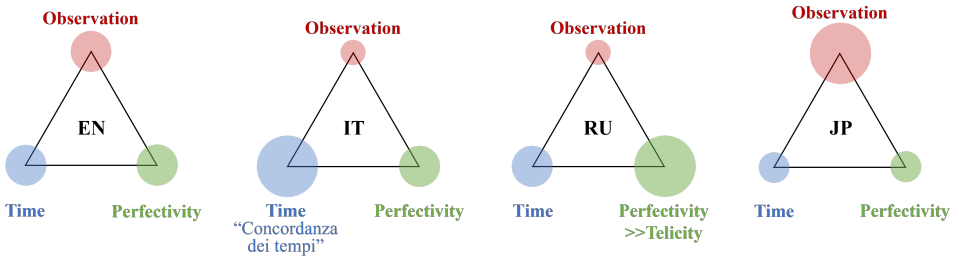


Figure 4.3: Schematic intuition: languages differ in how much temporal ordering and completion can be inferred from linguistic marking (tense/perfectivity) versus resolved from context (observation).

4.4.2 Dataset design: parallel templates for temporal relations and perfectivity

PERFECT TIMES is a multilingual multiple-choice temporal VideoQA benchmark with **3,739** question-answer pairs and a human gold standard of **93.36%**. Each item is built around two timestamped actions: a main-clause action (*MCA*, the queried action in the correct answer) and a dependent-clause action (*DCA*, the action that establishes temporal context). Questions are generated from parallel template families across languages, so that the same underlying temporal relation is queried with language-appropriate tense/aspect marking.

We generate questions by systematically varying:

- **Temporal relation** between MCA and DCA (precedence, succession, overlap), aligned to a broad relation inventory (Allen-style interval relations) [3];
- **Completion viewpoint** (completed vs. ongoing readings) in both MCA and DCA, to test sensitivity to perfectivity cues;
- **Distractor types** that separate temporal binding from action recognition (Section 4.4.4).

For readability, we summarize the design here and provide the full multilingual template inventory and examples in Appendix A.4.1 and Appendix A.4.2.

4.4.3 Template families and examples

Templates are organized into three families corresponding to common temporal queries in VideoQA: **precedence** (*before*), **succession** (*after*), and **simultaneity/overlap** (*while/when*). The goal is to force models to distinguish: event identity (*which action*), from event boundaries (*completed vs. ongoing*), and the relation demanded by the question.

Table 4.2: Examples of questions and answers in PERFECT TIMES generated by temporal and aspectual templates with respect to the telicity markers (t: telic, a: atelic). MCA: main clause action, DCA: dependent clause action.

Template	MCA	DCA	Question	Answer
			Precedence	
3	t	a	What had the person in the video done before holding the sandwich?	The person had opened the refrigerator.
4	t	t	What had the person in the video done before the other person walked through a doorway?	The person had closed the laptop.
6	a	a	What had the person in the video been doing before playing with a phone or camera?	The person had been watching the television.
7	a	t	What had the person in the video been doing before turning off the light?	The person had been playing with a phone or camera.
			Succession	
1	t	a	What did the person in the video do after tidying up the table?	The person put the food somewhere.
2	t	t	What did the person in the video do after they had opened the box?	The person put the bag somewhere.
10	a	a	What was the person in the video doing after sitting in a chair?	The person was sitting on the floor.
11	a	t	What was the person in the video doing after taking the cup from somewhere?	The person was drinking from the cup.
9	a	t	What was the person in the video doing when the other person closed the door?	The person was watching the television.
			Simultaneity	
5	t	a	What did the person in the video do while the other person was sitting on the floor?	The person opened the door.
8	a	a	What was the person in the video doing while holding the book?	The person was holding the bag.
12	t	t	What did the person in the video do when they grasped onto a doorknob?	The person opened the door.

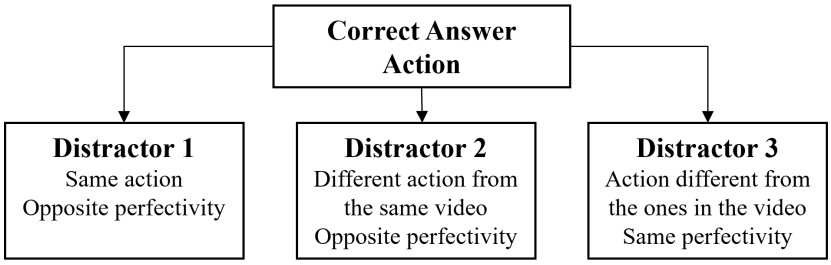


Figure 4.4: Distractor taxonomy in PERFECT TIMES relative to the correct answer.

4.4.4 Distractors: separating temporal binding from action recognition

Each item includes three distractors designed to probe distinct failure modes.

Distractor Type 1 represents the same action as in the correct answer, but with the opposite completion viewpoint. This directly tests sensitivity to perfectivity compatibility between question and answer.

Distractor Type 2 has a different action that occurs in the same video, and the same completion viewpoint manipulation as of the correct answer. It is paired with the same completed/ongoing manipulation as Type 1 and tests whether a model identifies the decisive temporal moment required by the question, not merely a plausible action.

Table 4.3: Illustrative minimal-pair structure for PERFECT TIMES. Type 1: same action, opposite completion viewpoint. Type 2: different in-video action with the same viewpoint manipulation.

Question	What **did** the person do **after** they **closed the box**?
Correct	They **picked up the tape**.
Type 1	They **were picking up the tape**.
Type 2	They **were putting the box down**.
Type 3	They **washed a plate**.

Distractor Type 3 denotes an out-of-context action. An action absent from the current video, serving as a basic grounding sanity check.

Option positions are randomized per item.

4.4.5 Concrete diagnostic example (minimal pair)

To make the distractor logic explicit, Table 4.3 illustrates a representative structure of one item. Type 1 preserves the same action but flips completion viewpoint; Type 2 replaces the action with another in-video event while applying the same flip.

4.4.6 Experimental protocol

We tested both the open-source and closed-source multilingual VLMs: Qwen2-VL [129], MiniCPM-V [140], InternVL2 [20], LLaVA-NeXT-Video [149], GPT-4o [91], and Gemini-2.0-Flash-Lite [120][1]. Additional details on open-source models are given in Table A.13 of Appendix A.5. The main metrics are Accuracy and Error Rate by distractor type. Model configurations and prompts are reported in Appendix A.5.

4.4.7 Main results: language sensitivity and distractor-driven failures

Table 4.4 reports the full multilingual results on PERFECT TIMES for all evaluated models, including the distractor-type error breakdown. This allows us to compare not only aggregate performance, but also the structure of temporal failures across models and languages.

The full results make two points clearer than the compact topline alone. First, performance varies substantially across models, confirming that temporal reasoning remains broadly unstable rather than failing only in one open/closed comparison. Second, the relative error mass on

[1]We use the experimental endpoint available via API.

Distractor Types 1 and 2 shows that most failures arise from imperfect temporal binding and completion compatibility, not from gross failures of visual grounding alone.

Most models ignore aspectual compatibility and fail under in-video confounds. Across models, the dominant error sources are Distractor Types 1 and 2, i.e., options that remain visually plausible and differ primarily in completion viewpoint and temporal placement. For most models, the majority of errors come from Type 2: a competing in-video action wins even when it does not match the temporal moment required by the question. This suggests that models often do not identify which boundary (start vs end, overlap vs succession) is decisive, even when they can recognize the relevant actions.

InternVL2 shows stronger temporal localization but weaker sensitivity to subtle completion contrasts. InternVL2 differs from the overall trend: its errors are dominated by Type 1 in every language except English. This indicates a model that is comparatively better at selecting the right *in-video event* (hence fewer Type 2 errors), yet still struggles to align the *completion viewpoint* expressed in the answer with that required by the question. Notably, InternVL2 performs worst in Italian, consistent with the idea that richer inflectional inventories for encoding temporal relations increase the burden on precise grammatical interpretation.

Finally, because questions are synchronized across languages, large language-wise deviations are diagnostically meaningful: if a system relied primarily on the video, accuracy should not swing strongly with the surface form. The comparatively stronger Russian performance across models is consistent with the hypothesis that VLMs partially recover temporality from lexical cues in the text (where completion contrasts can be especially salient), rather than binding completion to observation.

4.4.8 Implications for PCMD

PCMD requires exactly the capability that fails here: grounding a video into a temporally valid trace where step completion and ordering constraints are respected. The cross-linguistic results suggest that, in current VLMs, temporal reasoning is often extracted from language tokens (sometimes successfully; for instance, where lexical cues are strong), but does not reliably bind completion to observation.

This supports a practical design choice for PCMD: use VLMs where they are strongest — to produce accurate, fine-grained step interpretations (and the textual distinctions between correct vs incorrect step realizations), but perform the *verification decision* with a separate, explicit mechanism over step descriptions (sequence constraints, state consistency, edit struc-

ture), where temporal logic is auditable. This explicitly motivates the design of the HMM-based verifier in Chapter 6, which separates step recognition (VLM strength) from sequence logic (VLM weakness).

This motivates the two constructive parts of the thesis that follow. Chapter 5 addresses the data side by making completion-, ordering-, and recovery-sensitive deviations explicit in the procedural trace itself. Chapter 6 addresses the modeling side by separating step grounding from sequence-level verification, so that temporal logic is enforced by an explicit procedure model rather than inferred end-to-end by the VLM.

Table 4.4: VLM results on PERFECT TIMES across languages (percent). Columns **Distractor Type 1–3** report *error rates* by distractor type, i.e., the percentage of instances where the model selected an option of that type; therefore, **Acc + D1 + D2 + D3 = 100** for each row.

Model	Accuracy	F1 Macro	Distractor Type 1	Distractor Type 2	Distractor Type 3
English					
Gemini-2.0-flash-lite	43.41	42.19	22.15	30.42	4.01
GPT-4o	43.25	34.39	21.96	31.14	3.64
MiniCPM-V-2_6	36.19	35.5	27.68	31.08	5.05
Qwen2-VL-7B-Instruct	35.09	32.68	21.24	39.2	4.47
InternVL2-8B	34.86	33.36	27.34	29.09	8.7
LLaVA-NeXT-Video-7B	33.38	32.86	25.39	30.99	10.22
Italian					
Gemini-2.0-flash-lite	43.11	42.15	21.85	30.78	4.25
Qwen2-VL-7B-Instruct	41.59	39.63	21.95	32.09	4.35
GPT-4o	40.71	32.34	25.29	31.45	2.49
MiniCPM-V-2_6	37.42	35.73	29.55	28.96	4.07
InternVL2-8B	34.07	32.38	32.00	24.63	9.3
LLaVA-NeXT-Video-7B	25.92	17.65	25.40	29.74	18.88
Russian					
Gemini-2.0-flash-lite	46.99	46.33	19.04	29.85	4.12
GPT-4o	45.04	44.97	19.60	32.15	3.21
InternVL2-8B	36.87	34.37	28.82	25.54	8.77
MiniCPM-V-2_6	36.29	34.61	28.88	30.20	4.63
Qwen2-VL-7B-Instruct	34.13	32.75	20.40	39.5	5.96
LLaVA-NeXT-Video-7B	26.95	24.7	24.3	36.00	12.74
Japanese					
Gemini-2.0-flash-lite	43.06	41.77	22.19	31.43	3.32
MiniCPM-V-2_6	38.73	36.31	30.25	26.34	4.68
GPT-4o	38.49	30.65	23.78	35.23	2.49
Qwen2-VL-7B-Instruct	37.52	36.85	20.17	37.90	4.41
InternVL2-8B	35.05	31.65	31.22	23.92	9.81
LLaVA-NeXT-Video-7B	26.18	19.24	25.25	33.06	15.41

Chapter 5

PIE-V: Constructing and Benchmarking Mistake-Aware Procedural Videos

Real-world procedural assistants must account for *natural human errors* in tasks. However, such errors are underrepresented in instructional video datasets, and collecting dedicated mistake-aware datasets is costly, time-consuming, and human-resource intensive. Currently, only a few datasets explicitly include errors; their annotations are inconsistent, error taxonomies are idiosyncratic, and many "errors" are staged rather than spontaneous [47, 61, 96, 111]. As a result, large-scale realistic training remains a major bottleneck.

This chapter introduces **PIE-V** (**P**sychologically **I**nspired **E**rror injection for **V**ideos), a semi-synthetic framework for enriching clean procedures with *psychologically plausible* mistakes and *plausible recovery behavior*, together with a rubric for assessing mistake-aware datasets. PIE-V follows a *plan-first* approach: it decides *what* mistake happens *where*, and whether it is corrected, before rewriting the procedure text and optionally modifying the video.

Additionally, PIE-V takes into account that mistakes are often followed by recovery. In natural procedures, errors frequently trigger recovery: people notice an error and redo a step, insert a missing action, or undo an incorrect state [15, 16, 105, 106]. Many existing mistake-aware datasets either treat mistakes as isolated "anomalous segments" or do not model corrections explicitly, which can make them appear artificial and less representative of real-world procedural behavior. PIE-V explicitly generates **error–correction traces** to better approximate real executions where errors are embedded in coherent goal-directed activity.

Provenance. This chapter builds on the paper *How to Correctly Make Mistakes: A Framework for Constructing and Benchmarking Mistake-Aware Egocentric Procedural Videos* [74], accepted to the EgoVis Workshop at CVPR 2026. Related ideas were also presented in our

BMVA 2025 Vision & Language Symposium poster on Psychologically Inspired Error Injection in Instructional Videos.

Contributions in this chapter.

- A **dataset assessment rubric** targeting realism and coherence of mistakes at both step-level and procedure-level.
- A **plan-first enrichment pipeline** PIE-V that injects controlled errors and plausible corrections into clean procedures, guided by cognitive insights on error likelihood across procedure phases [15, 16, 88, 105].
- An empirical comparison between existing mistake-aware datasets, **LLM freeform** "make the procedure wrong" generation, and **PIE-V + LLM** generation.

Chapters 3 and 4 showed that PCMD cannot rely on end-to-end VLM judgment alone: correctness decisions must be grounded in explicit procedural structure, temporal dependencies, and stable verification signals. PIE-V responds to these findings on the data side. Its goal is not only to synthesize errors, but also to construct mistake-aware traces whose deviations, recoveries, and downstream consequences are explicit enough to support procedure-level verification.

5.1 Why current mistake-aware datasets are insufficient

Mistake-aware procedural datasets have been instrumental in establishing benchmark tasks for *procedural anomaly detection* and *mistake analysis*, where the primary goal is often to localize or classify visually observable deviations from the expected step execution at a given moment. Such benchmarks are tightly coupled to their task formulations and evaluation protocols (e.g., segment-level correctness vs. error) and are frequently designed to support error detection as an *isolated recognition problem* rather than full-horizon procedural reasoning [6, 63].

From the perspective of **procedure-aware mistake detection and correction**, these datasets are not "insufficient" per se; rather, they reflect sensible design choices aligned with their original objectives. However, PCMD places additional requirements on the data: the ability to represent *long-range dependencies* between steps, to relate mistakes to *downstream consequences*, and to capture the fact that real-world errors are frequently followed by *recovery actions* that restore the procedure to a valid state. In this chapter, we focus on these PCMD-specific requirements and analyze where current datasets provide partial coverage and where further enrichment is needed.

Beyond the raw size, for training and evaluation of realistic mistake detection and recovery the following systematic limitations matter:

(1) Small scale and staging. Only a small number of datasets include mistakes with step-level descriptions, and several rely on staged deviations rather than spontaneous ones. For instance, datasets may have few videos, narrow tasks, or experimental setups that differ from everyday environments, such as lab-like conditions, marked objects, scripted behaviors [47, 61, 96, 111].

(2) Missing recovery behavior. Errors in real procedures are often followed by recovery, such as redo, rollback, or compensatory actions. Ignoring recovery collapses a key aspect of human error management: whether an execution ends correct despite intermediate deviations [16, 105]. This omission also biases datasets toward looking "artificially wrong" rather than naturally imperfect.

(3) Taxonomy fragmentation and limited comparability. Datasets use different error taxonomies. They can be cooking- or assembly-specific classifications of errors or psychologically motivated by the nature of a mistake. Cross-dataset comparisons are difficult and solutions are not generalizable [47, 61, 96]. This thesis therefore emphasizes a unified structural taxonomy consistent with error-as-edit views: insertion, deletion, substitution, transposition, wrong execution and correction.

(4) Procedure-level coherence is not guaranteed. Even when step-level mistakes are labeled, the overall procedure may become incoherent with the text protocol or world state. As described in Chapter 2, this is especially relevant for PCMD: realistic evaluation requires modeling global sequence logic, causal consistency, and text–video alignment.

5.2 Rubric for assessing mistake-aware datasets

A central PIE-V contribution is a structured rubric for evaluating mistake-aware procedural datasets. The rubric targets both *local* realism of mistakes and *global* coherence of the whole procedure. It aligns with best practices for human evaluation where multiple complementary criteria are used rather than a single subjective score [4, 59].

Metric	Scale	What it measures
Error Validity	Binary	Whether the deviation should be considered a mistake or a plausible deviation.
Human Plausibility	Likert (1–5)	How natural and human-like the mistake appears in context (not "too perfect", not bizarre).
Confusability	Likert (1–5)	How difficult it is to notice the mistake (proxy for detectability / severity).
Procedure Logic	Binary + Likert (1–3)	Whether the overall procedure becomes logically broken due to the mistake(s), with annotator confidence.
Sequence Consistency	Likert (1–5)	Whether the edited step sequence remains consistent and executable as a procedure.
State-Change Coherence	Binary	Whether the world state implied by steps remains coherent (no impossible state transitions).
Video Plausibility	Likert (1–5)	Whether the video depiction of the mistake looks natural (when video is available).
Text–Video Grounding Consistency	Likert (1–5)	Whether the textual procedure matches what happens in the video at the episode level.

Table 5.1: Overview of the PIE-V dataset assessment rubric.

5.2.1 Rubric dimensions

Table 5.1 summarizes the main rubric dimensions used in our study. Full wording and annotator-facing descriptions are provided in Appendix B.2.

5.2.2 Annotation protocol and agreement

We use 5 annotators (2 male, 3 female; age 20–47; all with higher education). The annotation UI is shown in Figure 5.1.

To support consistent judgments, annotators are presented with a **paired setting**: (i) a **reference** execution of the same (or closely matched) procedure without mistakes (with video), and (ii) a **mistake-aware** execution containing **explicitly marked** mistakes and (when present) **marked corrections**. Thus, the task is *not* to discover where the mistake occurs, but to assess the *quality and realism* of the already indicated deviations and recoveries under the rubric.

For each rubric dimension, the interface provides a short metric description (a hint) explaining what the score should capture, and annotators select values from a drop-down menu (binary choices, categorical labels, or Likert scales depending on the metric, see details in Appendix B). This reduces ambiguity in interpretation and standardizes the rating procedure across datasets and generation settings.

Agreement is quantified with Krippendorff's α (Table 5.7). As expected, objective labels (e.g., Error Validity) show substantially higher agreement than inherently subjective ratings

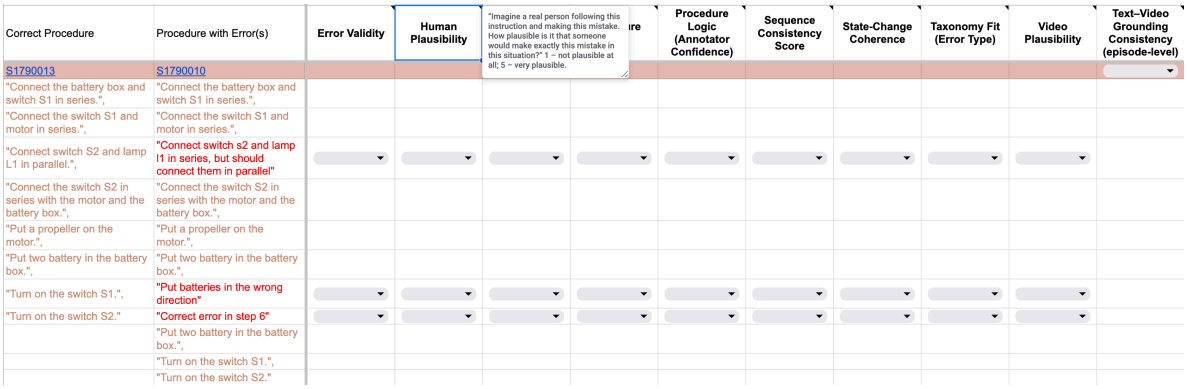


Figure 5.1: Annotation interface used for the PIE-V rubric (example from the electronics task in EgoOops). Annotators are shown a **reference** mistake-free execution (with video) alongside a **mistake-aware** execution containing **mistakes and corrections** (highlighted in red). The goal is not to localize errors but to **rate the indicated deviations** on step-level and procedure-level criteria. Each metric includes a short in-UI explanation, and ratings are selected via drop-down menus.

Dataset	Videos	Total steps	Mistake steps	Mistake rate	Avg. steps/video	Avg. mistakes/video
EgoPER	25	358	134	37.43%	14.32	5.36
EgoOops	25	302	87	28.81%	12.08	3.48
Assembly101	25	409	166	40.59%	16.36	6.64
CaptainCook4D	25	370	192	51.89%	14.80	7.68

Table 5.2: Audit sampling summary for the four existing mistake-aware datasets (25 videos each).

(e.g., Human Plausibility), consistent with prior findings on human evaluation reliability [4, 59].

5.3 Audit of four mistake-aware datasets

We apply the rubric to four established datasets with annotated mistakes: **EgoPER** [61], **EgoOops** [47], **CaptainCook4D** [96], and **Assembly101** [111]. For each dataset, we randomly sample **25 videos** per dataset that contain at least one annotated mistake. To contextualize the rubric scores, Table 5.2 reports the basic scale statistics of the audited subset (steps and mistake density), while Table 5.3 summarizes the aggregated rubric results. Full per-metric tables and additional breakdowns (including annotator-confirmed error subsets) are provided in Appendix B.

Table 5.2 shows that the audited subsets differ in step density and mistake density, which provides useful context for interpreting perceived realism and procedure-level coherence

Dataset	Err.Valid (Yes)	Human Pl.	Confus.	Proc.Logic (Yes)	Seq.Cons.	State-Chg (Yes)	Vid.Pl.	T–V Gr.
EgoPER	51%	2.67	1.88	26%	4.41	14%	3.22	3.42
EgoOops	72%	3.73	1.63	28%	4.50	10%	4.31	3.74
Assembly101	65%	3.69	2.09	12%	4.82	4%	4.05	3.22
CaptainCook4D	74%	3.71	2.32	12%	4.73	2%	3.42	3.63

Table 5.3: Aggregated rubric statistics for existing datasets. "Proc.Logic (Yes)" denotes the *rate of procedures judged logically broken* (lower is better). "T–V Gr." is Text–Video Grounding Consistency.

scores. With this context in mind, we report rubric scores in Table 5.3.

Summary of observations. Table 5.3 provides a compact snapshot of how existing mistake-aware datasets are perceived under our rubric.

These results should not be read as ranking. We use them to highlight *dataset-specific signatures* that matter for PCMD: how often annotated deviations are perceived as genuine mistakes (Error Validity), how human-like they appear (Human Plausibility), how noticeable they are (Confusability), how naturally they appear in the video (Video Plausibility) and how well textual protocols align with what is shown (Text–Video Grounding Consistency). These dimensions help explain why some datasets, despite being suited for step-level deviation recognition, offer only partial coverage for end-to-end mistake detection and correction.

EgoOops [47]. Across the evaluated sets, EgoOops is perceived as the most natural overall: it scores strongly on Human Plausibility and Video Plausibility, suggesting that mistakes tend to look less contrived and more behaviorally credible. At the same time, EgoOops is small in scale (50 videos across 5 tasks) and its scenarios are deliberately specific, such as chemistry-like setups and controlled assembly activities. The visual environment is also distinctive, including instrumented objects, such as QR-marked parts, which can reduce similarity to everyday procedures. As a result, despite strong realism cues, the dataset offers limited coverage for training and stress-testing PCMD in broad, real-world settings.

Assembly101 [111]. Assembly101 exhibits the lowest Text–Video Grounding Consistency (3.22). This indicates that the textual description of the procedure often diverges from what is actually executed in the video. This mismatch complicates learning and evaluation of procedure-level reasoning for multimodal models, where they must track which steps have truly occurred from step descriptions. Additionally, the task definition (assembling until completion) imposes structural constraints: some mistake types, such as genuine step omission, are naturally absent because the procedure continues until the assembly goal is met. On the

contrary, during assemblies multiple correct attachments and detachments may happen, which artificially increases the number of inserted in the procedure sequence steps.

EgoPER [61]. EgoPER stands out by having comparatively low Error Validity: only 51% of labelled deviations are judged by annotators as mistakes, while the remainder are perceived as plausible deviations. This suggests that, for a substantial fraction of cases, the boundary between "mistake" and "acceptable procedural variation" is ambiguous in practice. Such ambiguity is not necessarily a flaw of the dataset, but it is a key consideration for PCMD, where supervision signals should ideally distinguish recoverable deviations from true goal-threatening errors.

CaptainCook4D [96]. CaptainCook4D shows patterns consistent with high mistake density and weaker realism cues. Its Human Plausibility and Confusability profiles suggest that many deviations are easily noticeable and may appear staged, and the resulting error frequency per video is higher than what annotators typically expect for everyday cooking. This is consistent with a benchmark focus on detecting various anomalous execution segments, but it can reduce suitability as naturalistic training data for modeling realistic error rates and recovery dynamics.

Overall, these datasets were developed primarily to support *visual anomaly detection* at the level of single segments, rather than to serve as large-scale, coherent training material for procedure-level reasoning with temporal, causal, and recovery behavior. They also only partially reflect cognitive regularities of human errors (phase effects, detectability and recovery, realistic frequency of errors per procedure) emphasized in classical accounts for psychology of errors [16, 88, 105].

5.4 PIE-V pipeline: plan-first error and correction simulation

PIE-V is designed to close the above gaps by generating mistake-aware variants of clean procedural videos while preserving procedure coherence and incorporating plausible recovery. Figure 5.2 gives an overview of the pipeline.

5.4.1 Source procedures

We sample 50 clean procedures from the Ego-Exo4D [44] keystep annotations, covering all 17 keystep task families, such as bike repair, COVID test, cooking and more. Each procedure

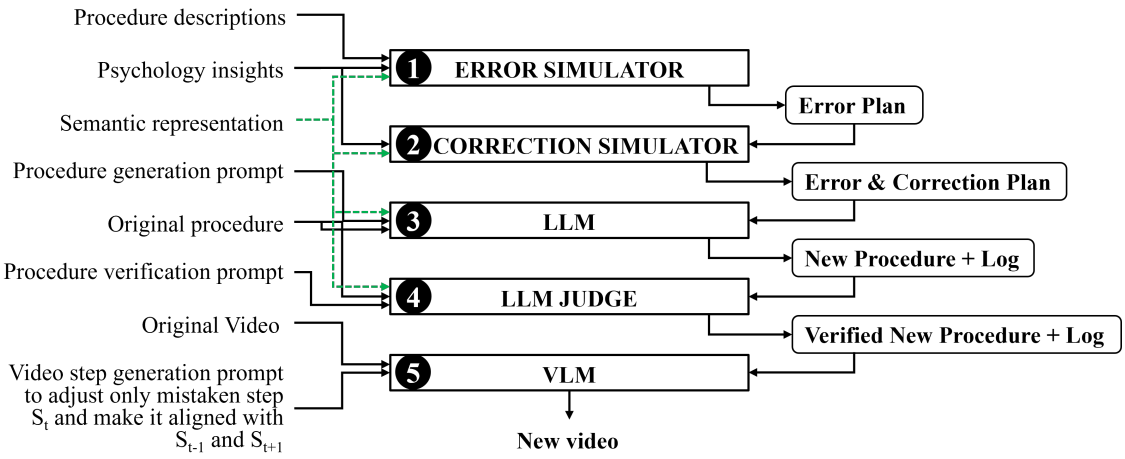


Figure 5.2: PIE-V pipeline overview. Clean keystep procedures are enriched by (1) an error planner (psychology-informed, constrained by step semantics and procedure phase), (2) a correction planner (recovery behavior), (3) an LLM writer (procedure rewriting with cascade consistency edits), (4) an LLM judge (coherence validation and repair; optionally multimodal), and (5) a video synthesis stage that generates new clips and smooth transitions for video plausibility.

has a unique step sequence. The sampled procedures range from short (minimum 3 steps) to typical mid-length procedures with the average of 25.18 steps.

5.4.2 Error simulator

PIE-V first generates an *error plan*: a set of error events specifying the target step index, error type, and (for execution edits) which semantic slot is perturbed. The simulator is grounded in cognitive accounts of procedural failures: error likelihood depends on the **phase** of a procedure and the **load/complexity** of steps, consistent with slip/lapse, memory overload theories, and post-completion vulnerability [15, 16, 88, 105].

Psychology-grounded error-type intuition. PIE-V uses an edit-based taxonomy (substitution, wrong execution, deletion, insertion, transposition), but the choice of where and which error to inject is further motivated by classical cognitive accounts as shown in Table 5.4. In particular, different error types are associated with different failure modes and tend to vary across procedure phases under changing cognitive load and fatigue:

Phase-conditioned priors and phase rates. Raw timestamp alone is too unstable because videos of the same task differ in duration and pacing. We partition each procedure by execution phase. PIE-V computes phases using a **step-load** signal combining (1) normalized

Error type	Key psychological interpretation and phase tendency
Substitution	Confusion between similar steps, associative interference, or schema competition; tends to be rarer in later phases once the execution pattern stabilizes.
Wrong execution	Execution slips and capture errors (local parameter mistakes), often due to unfamiliarity and motor learning early on; can occur throughout and may reappear toward the end under fatigue.
Deletion	Lapses due to memory overload and post-completion vulnerability; tends to increase over the procedure, often peaking toward the end as attention drops.
Insertion	Overgeneralization or associative activation (including unintended repetition); generally rarer, but can increase in the middle when multiple routines overlap and the performer is "in the flow".
Transposition	Sequencing/planning slips under high cognitive load; can occur under time pressure or fatigue, often when step ordering constraints are weak or when attention is divided.

Table 5.4: Qualitative cognitive motivations behind PIE-V error types and their phase tendencies (used to inform phase-conditioned priors).

step duration and (2) a semantic-complexity proxy (depth and richness of the step's semantic structure represented in semantic roles). Formally, the load of step i is:

$$\text{load}(i) = w_c \cdot \widehat{\text{complexity}}(i) + w_t \cdot \widehat{\text{duration}}(i).$$

PIE-V uses two complementary phase-level components: (i) a *phase error-rate model* that controls *where* errors are more likely to occur across the procedure, and (ii) *phase-conditioned type priors* that control *which* error types are more likely within each phase. The error-type order is

$$\texttt{ERROR\_TYPES\_ORDER} = [\texttt{wrong\_execution}, \texttt{deletion}, \texttt{substitution}, \texttt{insertion}, \texttt{transposition}],$$

and the phase-conditioned prior weights are:

$$\texttt{PHASE\_ERROR\_TYPE\_PRIORS} = \begin{cases} \text{phase\_1} : [3.5,\ 1.0,\ 2.5,\ 2.0,\ 1.0] \\ \text{phase\_2} : [2.0,\ 2.0,\ 1.5,\ 2.5,\ 2.0] \\ \text{phase\_3} : [3.5,\ 2.5,\ 1.0,\ 2.0,\ 1.0]. \end{cases}$$

Separately, the phase error-rate model (relative phase rates) is:

$$\texttt{PHASE\_ERROR\_RATE\_MODEL} = \{\text{phase\_1} : 0.10,\ \text{phase\_2} : 0.19,\ \text{phase\_3} : 0.14\},$$

which is normalized to multipliers with mean 1.0 so that the total number of errors remains governed by the procedure-level risk model, while redistributing error placement across phases.

Observed phase distribution in our split. Applying these settings to our 50-procedure sample creates the following realized counts (total 102 errors):

$$\text{Phase 1: } \{we : 10,\ i : 8,\ s : 10,\ d : 1\}$$
$$\text{Phase 2: } \{we : 11,\ i : 12,\ s : 11,\ d : 8,\ t : 6\}$$
$$\text{Phase 3: } \{we : 6,\ i : 8,\ s : 3,\ d : 5,\ t : 3\}.$$

These statistics match the intended phase trends, such as "omissions increase toward Phase 3", while still allowing variety and realism.

Semantic and ordering constraints. To avoid impossible or nonsensical perturbations, the simulator enforces constraints derived from step semantics and chronological dependencies. For example, transpositions that would violate basic object acquisition/use ordering are forbidden. More specifically, moving a "use tool" step before a "get tool" step for the same object is prohibited.

5.4.3 Correction simulator

Errors in real procedures are often noticed and corrected. PIE-V therefore generates an explicit *correction plan* conditioned on the error plan and procedure context. The correction simulator uses plausibility constraints and detection priors influenced by **error type** and **a severity/detectability proxy aligned with Confusability**. Thus, 27 correction events are generated on top of the 102 errors in our 50-procedure set.

Correction types. PIE-V models recovery actions grounded in cognitive research [14, 112, 123], such as: redoing the erroneous step, stopping and fixing immediately, undoing a state change and redoing, or rolling back and redoing when the error is discovered late. These choices reflect classic views where easily detectable slips are corrected more often than subtle or knowledge-based mistakes [105, 106].

5.5 PIE-V plans: examples from JSON logs

Below we show the full plan produced by PIE-V (errors + corrections), followed by one concrete rewritten procedure instance `cmu_bike14_4` for the "Install a Wheel" task in Subsection 5.5.1.

Planned error–correction trace (PIE-V plan).

```
[Step 01 / 5] (phase_1) ERROR: ‘‘Position the wheel correctly into the fork’’
-> wrong_execution
[Corr 01 after Step 01] (phase_1) CORRECTION for E01 wrong_execution: stop_and_fix
-> redo Step 01
[Step 02 / 5] (phase_2) OK: ‘‘Tighten the both axle nuts with hand’’
[Step 03 / 5] (phase_2) OK: ‘‘Get a wrench from the tool stand’’
[Step 04 / 5] (phase_2) ERROR: ‘‘Tighten the both axle nuts with a wrench’’
-> insertion
[Step 05 / 5] (phase_3) OK: ‘‘Put the wrench back in the tool stand’’
```

5.5.1 LLM writer: coherent procedure rewriting

Given the **original procedure**, the **error–correction plan**, and **step semantic representations**, PIE-V rewrites the procedure into a new textual trace that contains the planned deviations and recovery steps. Crucially, the writer performs **cascade adjustments** to preserve coherence: if an object is changed in an early step, later steps mentioning that object are rewritten accordingly (e.g., ingredient substitution propagates). This addresses a common LLM failure mode where local edits break global consistency.

We test two writers: **Qwen2.5-32B-Instruct** (open) and **GPT-5.2** (closed). Prompts and model identifiers are provided in Appendix B.

Example output (same plan, different writers). For the bike example above, GPT-5.2 produces:

```
Position the wheel slightly crooked into the fork.
Loosen the axle nuts, straighten the wheel in the fork,
and position the wheel correctly into the fork.
Tighten the both axle nuts with hand.
Get a wrench from the tool stand.
Set the wrench on the wrong axle nut size and it slips off.
```

```
Tighten the both axle nuts with a wrench.
Put the wrench back in the tool stand.
```

Qwen2.5 produces a different but related variant:

```
Position the wheel incorrectly into the fork.
Stop, fix the incorrect positioning, and position
the wheel correctly into the fork.
Tighten the both axle nuts with hand.
Get a wrench from the tool stand.
Tighten the both axle nuts with a wrench.
Spill some oil on the wrench while tightening the both axle nuts with a wrench.
Wipe the oil off the wrench before putting it back in the tool stand.
Put the wrench back in the tool stand.
```

5.5.2 LLM judge: validation, repair, and optional multimodality

The LLM judge checks the generated procedure for coherence, plausibility, and plan compliance. It revises incoherent outputs and rejects impossible mistakes. When repeated attempts fail internal validation, PIE-V optionally conditions the judge on a *mid-step frame* to ensure that the intended textual change remains compatible with the visual content. In our experiments, the judged setting uses: **GPT-5.2** for closed-source multimodal judging and **Qwen3-VL-32B-Instruct** for open multimodal judging.

Empirically, judge intervention was needed only for a small number of GPT-5.2 cases (6 procedures) and more for Qwen2.5 (17 procedures), suggesting that GPT-5.2 often produces plausible procedures on the first attempt. However, judging does not resolve all writer imperfections, so results are not yet perfect.

5.6 Video synthesis and smooth transitions

PIE-V as a textual & logical backbone. While PIE-V supports optional video editing and synthesis, its primary contribution is a textual and logical backbone for mistake-aware procedural data: plan-first error placement, recovery modeling, and globally coherent procedure rewriting. Video generation is therefore treated as an optional rendering layer that can visualize the planned edits, but its fidelity is ultimately bounded by the chosen video model, reference conditioning, and stitching artifacts. Accordingly, in this thesis we emphasize the

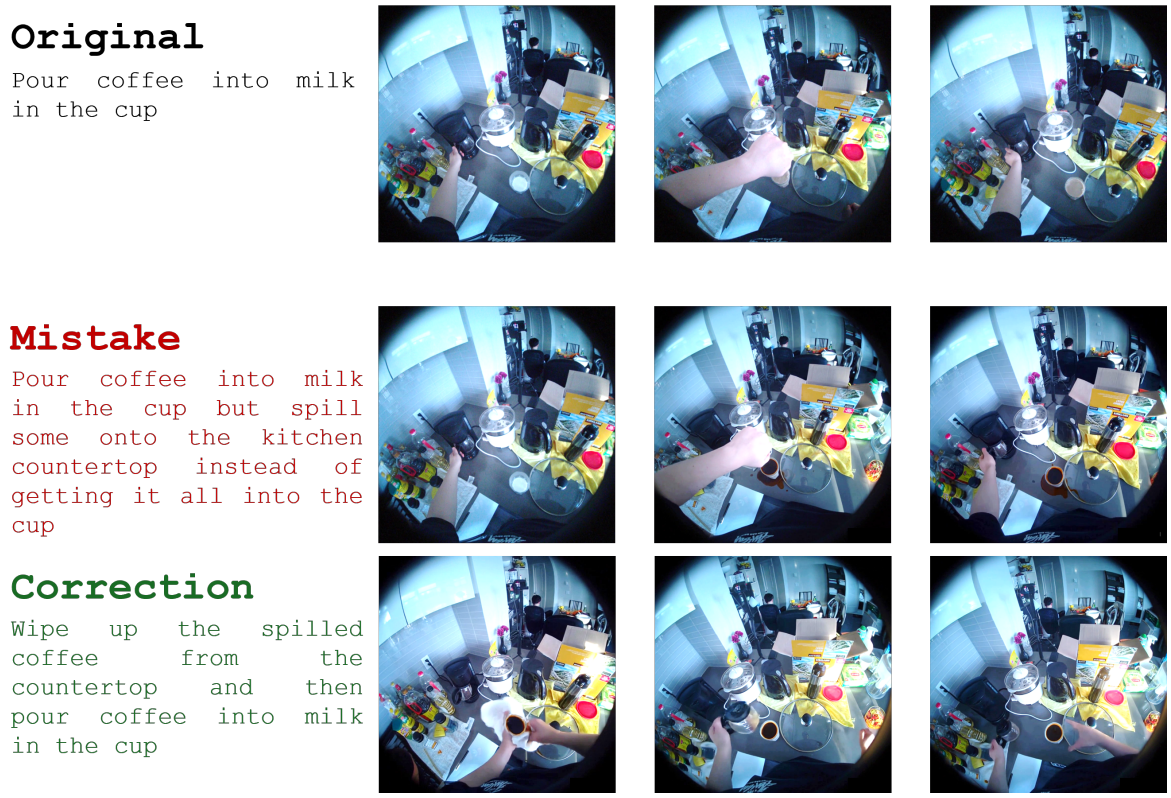


Figure 5.3: Coffee example (Making Coffee latte, **sfu_cooking_008_5**): (A) Original step: the person takes the coffee jar, pours coffee into the cup, and puts the jar back; (B) Generated wrong execution step: the person takes the coffee jar, pours too much coffee into the cup and spills while pouring, puts the jar back; and (C) Generated correction: the person wipes out the small pool on the countertop with a paper towel, pours more coffee in the cup and puts the jar back.

controllable plan and the resulting procedure-level coherence, and use video examples mainly as qualitative illustrations of how the planned error–correction traces can be realized visually.

In our implementation, PIE-V goes beyond text by synthesizing new video clips for modified/inserted/correction steps and by smoothing transitions to maintain high **Video Plausibility**. We use a video model **Kling-O** [121] to generate short clips conditioned on **physically grounded prompts**, **reference frames**, and **optional feature references**. For smoothness, we enforce that the state at the end of clip A matches the state at the start of clip B, and we stitch clips with crossfades of tuned duration. Concretely (see scripts in Appendix B), we set the first frame of B to be the last frame of A, and we use slightly longer crossfade for the critical error→correction boundary. This makes the A→B continuity highly reliable under consistent prompts and reference frames.

For the procedure **sfu_cooking_008_5** (Making Coffee latte), PIE-V planned a wrong exe-

cution mistake at the final pour step and a rollback-and-redo correction. The generated steps include a visible spill and a subsequent cleanup+redo:

```
...
Pour coffee into milk in the cup but spill some onto the kitchen countertop
instead of getting it all into the cup.
Wipe up the spilled coffee from the countertop
and then pour coffee into milk in the cup.
```

The generated video clips along with the correct step are shown in Figure 5.3.

5.7 Baselines: freeform LLM "make it wrong" generation

A natural question is whether such a structured pipeline is necessary: perhaps an LLM can generate plausible mistakes implicitly from its training data. We therefore test a **freeform** setting where the LLM receives a clean procedure and is asked to introduce 1–3 plausible mistakes without any explicit plan.

This task is unexpectedly difficult for LLMs, because instruction-following models are inherently optimized to *avoid* errors. In practice, models often "play safe" by deleting a final step (creating the appearance of post-completion errors), but struggle with consistent multi-step object tracking and long-horizon coherence, especially for very long procedures. In our 50-procedure sample, a randomly included very long cooking procedure (130 steps) proved too challenging for both models to generate a coherent mistaken variant in any setting.

5.8 Comparison: existing datasets vs freeform LLM vs PIE-V+LLM

Table 5.5 compares aggregate rubric statistics for the freeform LLM generation and PIE-V+LLM (judged) generation. Video-related metrics are not reported in the summary tables for text-only generations, as videos were generated only for the best setting, Ego-Exo4D-GPT-5.2-PJ.

Key comparison outcomes. Table 5.5 highlights a consistent pattern across settings. First, moving from freeform generation to a plan-conditioned and judged pipeline improves both perceived realism and procedural reliability. Compared to freeform GPT-5.2, the PIE-V+GPT-5.2 judged setting increases the fraction of agreed mistakes (Error Validity: 89% vs. 83%) and

Dataset	Err.Valid (Yes)	Human Pl.	Confus.	Proc.Logic (Yes)	Seq.Cons.	State-Chg (Yes)
Ego-Exo4D-Qwen (freeform)	57%	3.22	2.02	36%	4.20	27%
Ego-Exo4D-GPT-5.2 (freeform)	83%	3.69	2.02	25%	4.81	7%
Ego-Exo4D-Qwen-PJ (PIE-V+Qwen2.5, Qwen3-VL-judged)	71%	3.17	1.41	30%	4.48	10%
Ego-Exo4D-GPT-5.2-PJ (PIE-V+GPT-5.2, judged)	89%	3.83	1.82	6%	4.91	3%

Table 5.5: Aggregate rubric statistics for synthetic generations. "PJ" denotes PIE-V + Judge. Lower "Proc.Logic (Yes)" is better.

Dataset	Human Pl.	Confus.	Proc.Logic	Seq.Cons.	State-Chg	Vid.Pl.	T–V Gr.
EgoPER	2.24	1.66	Yes (40%), No (60%)	2.68	Yes (21%), No (79%)	2.92	3.42
EgoOops	3.20	1.64	Yes (33%), No (67%)	2.67	Yes (11%), No (89%)	4.29	3.74
Assembly101	3.78	2.24	Yes (16%), No (84%)	2.81	Yes (5%), No (95%)	4.11	3.22
CaptainCook4D	3.63	2.28	Yes (14%), No (86%)	2.74	Yes (8%), No (92%)	3.84	3.63
Ego-Exo4D-Qwen (freeform)	2.96	1.81	Yes (40%), No (60%)	2.59	Yes (30%), No (70%)	–	–
Ego-Exo4D-GPT-5.2 (freeform)	3.14	2.23	Yes (23%), No (77%)	2.74	Yes (14%), No (86%)	–	–
Ego-Exo4D-Qwen-PJ (PIE-V+Qwen2.5, Qwen3-VL-judged)	3.17	1.39	Yes (30%), No (70%)	4.48	Yes (11%), No (89%)	–	–
Ego-Exo4D-GPT-5.2-PJ (PIE-V+GPT, judged)	3.78	1.79	Yes (7%), No (93%)	4.91	Yes (2%), No (98%)	3.93	4.76

Table 5.6: YES-only statistics: computed only over deviations for which a majority of annotators agreed that the deviation is a mistake (Error Validity = Yes).

slightly improves Human Plausibility (3.83 vs. 3.69), while also lowering Confusability (1.82 vs. 2.02), indicating that mistakes become less borderline and easier for annotators to identify as genuine mistakes.

This trend is further reflected in inter-annotator agreement. In Table 5.7, Krippendorff's α for Error Validity is substantially higher for PIE-V+GPT-5.2-PJ ($\alpha = 0.913$) than for freeform GPT-5.2 ($\alpha = 0.701$). Intuitively, the plan-first generation and subsequent judging reduce ambiguous "could be a deviation" cases by enforcing a clearer error specification and recovery behavior.

In contrast, Qwen in the PIE-V setting frequently violates the intended error schema in practice: it may change the error type (e.g., implicitly swapping steps instead of performing the prescribed insertion), underperform cascade consistency edits, and omit prescribed corrections. Because this plan non-compliance undermines the validity of the PIE-V setting itself, we treat PIE-V+Qwen-PJ as a non-validated configuration (due to plan non-compliance) and do not treat it as a final product-quality setting, even though we report its rubric statistics for completeness.

To separate overall generation quality from disagreement about whether a deviation counts as a genuine mistake at all, Table 5.6 reports the rubric statistics after restricting the analysis to cases where a majority of annotators marked Error Validity = Yes.

The YES-only view separates borderline or disputed deviations from cases that annotators actually agree are mistakes. In practice, this makes it easier to compare realism and coherence conditional on the sample being accepted as a genuine procedural error.

Finally, we report Krippendorff's alpha across all metrics and settings (Table 5.7), which

Dataset	Err.Valid α	Human Pl. α	Confus. α	Proc.Logic α	Seq.Cons. α	Taxonomy Fit α
EgoPER	0.912	0.541	0.368	0.728	0.628	0.759
EgoOops	0.916	0.592	0.375	0.836	0.667	0.882
CaptainCook4D	0.694	0.758	0.847	0.621	0.542	0.791
Assembly101	0.859	0.584	0.697	0.739	0.649	0.931
Ego-Exo4D-Qwen (freeform)	0.568	0.539	0.417	0.579	0.543	0.820
Ego-Exo4D-GPT-5.2 (freeform)	0.701	0.424	0.341	0.593	0.652	0.752
Ego-Exo4D-Qwen-PJ (PIE-V+Qwen2.5, Qwen3-VL-judged)	0.958	0.483	0.358	0.631	0.706	0.905
Ego-Exo4D-GPT-5.2-PJ (PIE-V+GPT, judged)	0.913	0.489	0.387	0.672	0.619	0.803

Table 5.7: Krippendorff's α agreement summary for core rubric dimensions. The full set of metrics (including video-related columns) is reported in Appendix B.

provides a complementary view of how subjective or controversial each metric is under different data-generation regimes.

5.9 Discussion and limitations

Why plan-first helps. Cognitive accounts predict structured phase effects (e.g., post-completion omissions) and correlations between detectability and recovery. A freeform LLM has difficulty explicitly controlling these properties, especially in long procedures and when global object/state consistency must be maintained. PIE-V makes these constraints explicit and thus supports controllable generation and better auditability.

Limitations. PIE-V is not yet a fully automated "push-button" dataset builder. The pipeline depends on the availability and quality of step annotations and on the reliability of LLM rewriting and judging. Video synthesis quality varies with reference conditioning and prompt sensitivity, and the judged stage does not guarantee perfect plan compliance.

Why PIE-V is not yet a direct PCMD benchmark for HMM-based modeling. PIE-V is designed primarily as a pipeline for generating realistic mistake-aware procedural traces. The 50 source procedures in this thesis are intentionally diverse and mostly unique at the level of full step sequence, because the goal is to validate controllable mistake injection across heterogeneous procedures. For a statistical sequence model such as an HMM, one correct exemplar per canonical procedure would mainly test memorization of that exemplar. A benchmark for ChronoFix-style modeling introduced in the following chapter would instead require multiple correct executions per canonical task together with mistake-enriched variants derived from them.

Summary. This chapter introduced PIE-V, a semi-synthetic framework for mistake-aware procedural data. PIE-V combines: (1) a rubric for evaluating realism, coherence, and (when available) text–video grounding; and (2) a plan-first pipeline that injects controlled, semantics-aware errors and plausible corrections into clean procedures. The following chapter reuses the same instruction-grounded, edit-based view of procedural deviation at the modeling level.

Chapter 6

ChronoFix: Language-Grounded Post-Completion Mistake Detection in Procedural Videos

We study post-completion mistake detection in procedural videos: given a fully observed procedural execution and a reference procedure, the system must recover the executed step sequence and detect deviations from a valid reference trace. This chapter unifies two complementary lines of work. The first introduced the HMM-based ChronoFix backbone and validated it on CaptainCook4D and Assembly101 as a structure-first alternative to vision-only error detection. The second extended the language interface of the model, comparing raw step descriptions, semantic role representations, and action–object abstractions under a common protocol on CaptainCook4D, EgoPER, and EgoOops. Taken together, these studies support the same claim: PCMD is best solved in language space, where mistakes appear as deviations in step content and order and can be verified by an explicit sequence model.

To this end, we present ChronoFix, a language-grounded framework that maps segmented video clips to step descriptions, optionally normalizes them into structured semantic representations, and detects mistakes with a lightweight Hidden Markov Model over step sequences. The contribution is not HMMs in isolation, but a decomposition: perception produces grounded step evidence, while verification is performed by an explicit procedure model that is interpretable, data-efficient, and portable across benchmarks.

Provenance. This chapter draws on two manuscripts: *Post-Completion Mistake Detection in Task Structures of Procedural Video*, which introduced the HMM-based PCMD formulation and the CaptainCook4D/Assembly101 experiments, and *Mistake Detection in Proce-*

dural Videos via Language-Grounded Step Reasoning, currently under review for COLM 2026, which extended the framework with language-grounded step representations and cross-benchmark evaluation on CaptainCook4D, EgoPER, and EgoOops.

Conceptually, this chapter complements Chapter 5. There, the thesis addressed the data bottleneck by constructing more realistic mistake-aware traces; here, it addresses the inference bottleneck by verifying grounded step sequences with an explicit procedure model.

6.1 Why decomposing perception and procedure helps

Procedural videos are an attractive target for automated quality control: the underlying activity is structured, outcomes can be safety-critical, and mistakes are often detectable because they violate an expected temporal program. Yet most modern approaches attempt to solve the full problem end-to-end in vision space [34, 49, 56, 79, 151], training large models to directly classify mistakes from visual cues. This is compelling research, but it is also expensive, data-hungry, and often difficult to interpret.

Our position is that **procedural video is fundamentally a task-structure problem.** Vision matters, but mostly as a *front-end* that produces an execution trace. Once we have a reasonable textual trace (steps, actions, objects), the remaining problem, i.e. tracking progress in a canonical procedure and detecting deviations, is well matched to lightweight probabilistic models. This decomposition is attractive for three reasons:

- **Efficiency and portability.** We can reuse strong pretrained VLMs (e.g., Gemini-class systems) to convert video to a step-like textual trace, and then run a small, transparent probabilistic model such as ChronoFix for structure inference. This avoids training a large vision model for every new domain.

- **Interpretability.** A probabilistic state over canonical steps and explicit edit-style deviations provide human-readable diagnostics: where the agent is in the recipe, what went wrong, and how recovery happened.

- **Modularity.** Better perception immediately improves the system without changing the task model, while better task structure (scripts, priors, constraints) improves robustness even when perception is noisy.

6.2 Problem Setup: Post-Completion Mistake Detection under a Unified Edit-Based Taxonomy

We assume a task has a **canonical procedure** represented as an ordered list of steps $\mathcal{S} = (s_1, \ldots, s_N)$, where each step is a short textual description (e.g., "crack eggs"). At test time we observe an **execution trace** $\mathbf{o}_{1:T} = (o_1, \ldots, o_T)$, where each observation corresponds to a time segment and is represented textually. This trace may come directly from text instructions (robot logs, annotations) or from a VLM that summarizes video segments into step-like descriptions.

What makes PCMD different from "anomaly detection". Unlike generic anomaly detection, PCMD requires the model to maintain a coherent belief about *where the agent is* in a canonical procedure and *how to return* to that procedure after a deviation. Concretely, PCMD decomposes into three coupled subproblems: (i) **progress tracking** (which canonical step is being executed), (ii) **deviation detection** (omissions, insertions, substitutions, transpositions), and (iii) **recovery recognition** (returning to the intended trajectory after a deviation).

Error patterns in real datasets. Figure 6.1 illustrates representative deviations from CaptainCook4D and Assembly101. We highlight two common families. **Sequence errors** alter the global temporal program (e.g., out-of-order steps or extra steps). **Step execution errors** are local failures within an otherwise correct step (e.g., an incorrect technique or wrong part orientation). When available, we report step-level execution-error labels separately from structural sequence edits (see Section 6.7).

Edit-based formulation. Throughout this chapter, we interpret structural mistakes as an edit script that transforms the inferred step sequence into a valid reference procedure: **insertions**, **deletions**, **substitutions**, and **transpositions**. This view lets us map dataset-specific labels into a shared structural space and evaluate deviation detection under a common protocol described in Section 6.7. Recovery remains conceptually important, but in this chapter it is analyzed qualitatively rather than scored as a primary metric.

Unified label space. In the unified evaluation, labels from CaptainCook4D, EgoPER, and EgoOops are mapped into the four structural categories above. Assembly101 is kept outside this shared leaderboard and is discussed separately as an auxiliary transfer setting. When a benchmark also provides finer execution-error tags, we use them only in benchmark-specific discussion and do not merge them into the four-way leaderboard.

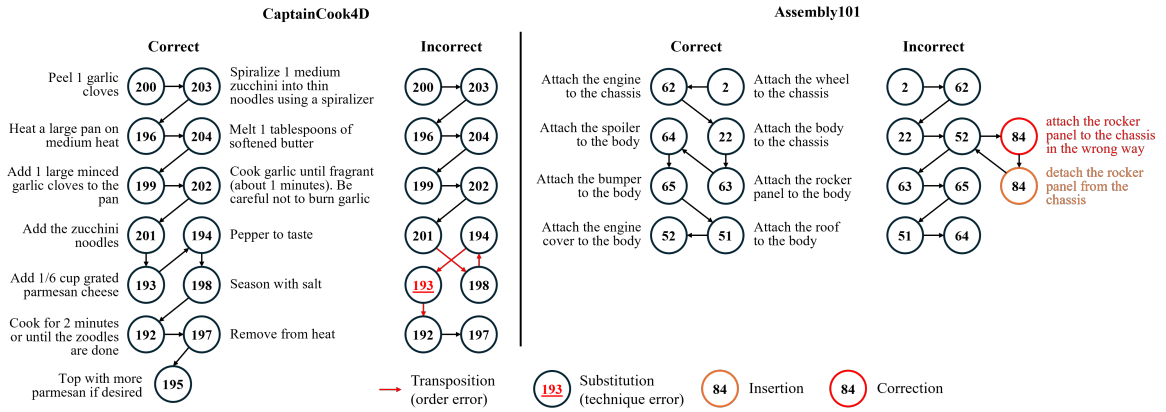


Figure 6.1: Correct vs. erroneous procedural executions in CaptainCook4D (left) and Assembly101 (right). Nodes are canonical STEP_IDS. "Incorrect" graphs visualize deviations from the reference. Red marks **sequence errors** (order/extra steps); colored nodes mark **step execution errors** (local failures within a step). Short step descriptions are shown for interpretability.

6.3 CHRONOFIX

CHRONOFIX proposes a multi-stage PCMD pipeline, summarized in Figure 6.2. First, we train Gaussian HMMs on step embeddings aggregated from correct executions. At inference, these HMMs decode the optimal step sequence for a test video, which is then aligned against a reference procedure to identify structural errors via edit operations.

6.4 CHRONOFIX Backbone: HMM for Canonical Progress

6.4.1 HMM Training

For every video we first get the step ID, timestamps, and textual description. Then we compute embeddings at the step level with a visual, textual or multimodal encoder. We denote the sequence of embeddings as $x_{1:T} = (x_1, \ldots, x_T), \quad x_t \in \mathbb{R}^{D_0}$. We apply PCA for dimensionality reduction to avoid overfitting and get the regularized embedding $\mathbf{z}_t = \mathrm{PCA}(\mathbf{x}_t)$.

To improve robustness, we add Gaussian jitter to a random subset of embeddings:

$$\mathbf{z}_t \leftarrow \mathbf{z}_t + \mathcal{N}(0, \sigma^2 I). \tag{6.1}$$

We model each task with the HMM $S = \{1, \ldots, K\}$, where S is the set of discrete step IDs as states corresponding to the step embeddings $\mathbf{z}_{1:T}$ as observations. The HMM parameters are

$$\theta_{\text{task}} = \big(\boldsymbol{\pi},\ A,\ \{(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s)\}_{s \in S}\big), \tag{6.2}$$

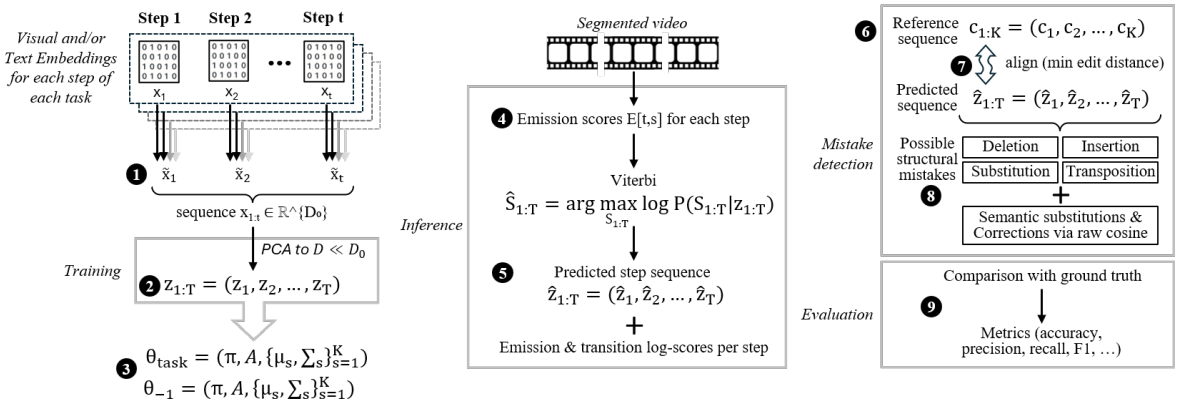


Figure 6.2: PCMD pipeline. For each step of each training video, we obtain a textual, visual, or multimodal encoding and average it across all embeddings of all steps with the same ID, $\tilde{x}_i$ (1). After PCA regularization, we obtain the representation of each step, z_i (2). During training, the model θ_{task} learns sequences of z_i for each task. Additionally, when we put all tasks together, we obtain a global HMM $\theta_{\text{-1}}$ (3). At inference time, we obtain embeddings for each video segment in the same way as during training and compute the emissions $E[t, s]$ (4). We then run Viterbi for sequence inference and keep the emission and transition scores for each step (5). For each task, we also have a set of possible non-erroneous sequences (6). We align the predicted sequence to the reference sequence using the minimal edit distance (7). Mismatches in the alignment indicate possible structural errors. Falling below the predefined thresholds for emission and transition scores indicates possible substitutions within a step (8). During evaluation, we compare the predicted errors with the ground-truth error annotations of the dataset (9).

where

$\boldsymbol{\pi} \in \mathbb{R}^K$ is the initial state distribution over S ($\pi_s = P(S_1 = s)$),

$A \in \mathbb{R}^{K \times K}$ is the transition matrix with entries $A_{ij} = P(S_t = j \mid S_{t-1} = i)$,

$(\boldsymbol{\mu}_s, \boldsymbol{\Sigma}_s)$ are the mean and covariance of the Gaussian emission for state $s \in S$.

The state mean is estimated as

$$\boldsymbol{\mu}_s = \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{z}^{(i)}, \tag{6.3}$$

where n_s is the number of training embeddings assigned to step s, and $\mathbf{z}^{(i)} \in \mathbb{R}^D$ is the i-th representation vector for that step after preprocessing.

For each state, the Gaussian emission model is estimated from the representation vectors assigned to that state. To stabilize estimation under limited data, we apply Ledoit–Wolf shrinkage and retain the resulting diagonal covariance at inference time:

$$\Sigma_s = \operatorname{diag}\left(\sigma_{s,1}^2, \ldots, \sigma_{s,D}^2\right). \tag{6.4}$$

This keeps the emissions numerically robust without introducing unreliable cross-feature correlations.

For the transition model we count how often each transition $s \to s'$ is observed and apply Laplace smoothing β:

$$A_{ss'} = \frac{c_{ss'} + \beta}{\sum_k (c_{sk} + \beta)}. \tag{6.5}$$

The smoothing ensures that every possible transition has a nonzero probability which is needed for rare transitions and unseen (erroneous) steps at test time.

We then enforce a near-diagonal structure reflecting the mostly sequential nature of the procedures: transitions that jump too far in step index are suppressed and the rows are renormalized.

For the initial distribution, we count how many correct videos start from each step s and apply the same smoothing.

In addition to per-task HMMs we train a global model θ_{-1} by pooling data across all tasks. This model is used as a fallback when a task has too few examples.

During training we also record calibration statistics (mean and quantiles) of emission scores and transition log-probabilities on correct videos. For Gaussian emissions, the emission score is the log-likelihood in Equation (6.6); for cosine emissions, it is the temperature-scaled cosine score in Equation (6.8).

$$\begin{aligned} \log p(\mathbf{z}_t \mid S_t = s) = -\frac{1}{2}\big[&(\mathbf{z}_t - \boldsymbol{\mu}_s)^\top \boldsymbol{\Sigma}_s^{-1} (\mathbf{z}_t - \boldsymbol{\mu}_s) \\ &+ \log|\boldsymbol{\Sigma}_s| + D\log(2\pi)\big]. \end{aligned} \tag{6.6}$$

Here the term $(z_t - \boldsymbol{\mu}_s)^\top \boldsymbol{\Sigma}_s^{-1} (z_t - \boldsymbol{\mu}_s)$ is the squared Mahalanobis distance between z_t and the state mean. The term $\log|\boldsymbol{\Sigma}_s|$ penalizes volume (uncertainty) of the Gaussian, and $D\log(2\pi)$ is a normalization constant ensuring that $p(\cdot \mid S_t = s)$ integrates to 1.

These statistics later inform thresholds for low-probability events.

6.4.2 HMM Inference

At test time, given a new video, we process every executed step exactly as in training: we extract a step-level feature $\mathbf{x}_t$, apply the same normalization and PCA projection to obtain $\mathbf{z}_t$, and select the HMM for the corresponding task (or the global fallback).

For Gaussian emissions we compute

$$E_{t,k} = \log \mathcal{N}(z_t; \mu_k, \Sigma_k). \tag{6.7}$$

For cosine emissions we compute

$$E_{t,k} = T \cdot \cos(z_t, \mu_k), \tag{6.8}$$

with temperature $T > 0$. This matches the geometry of normalized CLIP-like features and can be seen as a logit for a softmax over states. We L2-normalize vectors before cosine to ensure a bounded, temperature-scaled score.

We also support a global emission scaling factor α:

$$\tilde{E}_{t,k} = \alpha \cdot E_{t,k}, \tag{6.9}$$

which adjusts the relative weight between emission and transition terms in the Viterbi objective.

We then use the Viterbi[1] algorithm to decode the most likely step sequence:

$$\hat{S}_{1:T} = \arg\max_{S_{1:T}} \log P(S_{1:T} \mid \mathbf{z}_{1:T}) \tag{6.10}$$

In our setting, the Viterbi score of a candidate path $S_{1:T}$ is

$$\begin{aligned} \log P(S_{1:T}, \mathbf{z}_{1:T}) = \log \pi_{S_1} + \sum_{t=2}^{T} \log A_{S_{t-1},S_t} \\ + \sum_{t=1}^{T} \log p(\mathbf{z}_t \mid S_t). \end{aligned} \tag{6.11}$$

i.e., a sum of start, transition, and emission log-probabilities.

After predicting the sequence we record, for each time step t, the emission score and transition log-probability along the Viterbi path:

$$e_t^{\text{em}} = \tilde{E}_{t,\hat{S}_t}, \qquad e_t^{\text{tr}} = \log A_{\hat{S}_{t-1},\hat{S}_t}, \tag{6.12}$$

and refer to $(e_t^{\text{em}}, e_t^{\text{tr}})$ as the HMM path scores at step t.

[1]Viterbi is a dynamic-programming algorithm for HMMs that computes the maximum a posteriori (MAP) sequence of hidden states and guarantees to find the optimal one.

6.5 From Decoded Step Paths to Mistake Events

The HMM backbone gives a decoded step path $\hat{S}_{1:T}$ and path-wise scores. We convert this output into discrete mistake events by aligning the decoded path to the closest valid reference sequence and by using low emission or transition scores as generic anomaly signals when a deviation cannot be cleanly assigned to an edit type.

Each procedure, regardless of its specific execution in the dataset, has one or more valid correct step sequences. This is an abstract representation of the different ways to successfully complete the task. The pool of such reference sequences is formed either from the corresponding written instructions provided by annotation guidelines, from error-free videos, or by reconstructing them from videos with mistakes, removing the erroneous steps. Let a reference sequence of a given procedure be $c_{1:K} = (c_1, \ldots, c_K)$, where each c_k is a step ID.

Next we compare the sequence predicted by the algorithm $\hat{S}_{1:T}$ with one of the reference sequences, namely, the closest reference variant by edit distance[2]. We align the predicted sequence to the chosen reference sequence using global sequence alignment (Needleman–Wunsch or difflib[3]).

6.5.1 Edit operations on step sequences

The alignment of the inferred step sequence $\hat{S}_{1:T}$ to the selected reference sequence $c_{1:K}$ reveals structural error types:

- **Deletion:** a canonical step is missing in the inferred trace.
- **Insertion:** an inferred step does not correspond to any canonical step at that position.
- **Substitution:** a step is present but mismatched.
- **Transposition:** in canonical and predicted sequences steps occur in a different order (technically, it is both Insertion and Deletion of the same step in different positions).

In addition to edit-distance-inferred errors, we use the HMM path scores from Equation (6.12). Their negated versions $s_t^{\text{em}} = -e_t^{\text{em}}$ and $s_t^{\text{tr}} = -e_t^{\text{tr}}$ serve as auxiliary anomaly

[2]For better alignment, in assembly scenarios we first collapse duplicates keeping the last occurrence of each step ID due to the multiple extra attaching/detaching cycles.

[3]Needleman–Wunsch is a classical dynamic-programming algorithm for global sequence alignment. It optimizes a well-defined objective and is preferable when we need a full control over substitution and gap penalties. Python's difflib.SequenceMatcher is a heuristic matcher that focuses on maximizing the length of matching blocks. It is often more forgiving and may produce alignments that match human intuition when sequences share long common subsequences but differ locally. It is fast and works well as a robust default when we do not want to carefully tune alignment costs. In practice, we use difflib as the default alignment mode.

scores at step t. Given thresholds τ_{em} and τ_{tr} derived from correct validation videos, we flag a low-emission event if $s_t^{\text{em}} > \tau_{\text{em}}$ and a low-transition event if $s_t^{\text{tr}} > \tau_{\text{tr}}$. These are auxiliary diagnostic flags, not extra classes in the unified four-way evaluation.

We also compute an auxiliary within-state atypicality signal based on cosine similarity between a step embedding and the mean embedding of its assigned HMM state. Low cosine indicates that the content of the step is atypical even when the decoded discrete step ID remains unchanged. We use this as a diagnostic signal only.

6.5.2 Return to a valid trajectory

When a dataset explicitly contains corrective or recovery-like steps, we analyze whether the decoded path returns to a high-probability region consistent with the reference order after a flagged deviation. In the present chapter, however, recovery is treated as a secondary qualitative signal.

6.6 Language grounding and step representations

We use language as the interface between video and procedural reasoning. We report runs on the benchmark step descriptions as an upper bound. These text-only runs do not use video. In the practical setting, a VLM maps each pre-segmented clip to text from the task vocabulary, and the HMM operates on embeddings of that grounded text.

Raw step descriptions. We start from the canonical step descriptions provided by each benchmark. For erroneous steps, we use the benchmark-specific observed or modified step descriptions when they are available. When a benchmark provides only a short error cue instead of a full observed step, we convert it into a complete modified description by combining the canonical step text with the error cue. We also apply minimal text cleanup to remove benchmark-specific artifacts that do not carry procedural meaning. Broader dataset context is summarized in Appendix B, Section B.1.

The resulting representation keeps the lexical detail of the original annotation, including quantities, destinations, surfaces, and local modifiers. That detail matters because many procedural mistakes differ from the canonical step in terms of the arguments, not only in terms of the main verb.

Semantic role representations. We normalize each step description into a predicate–argument form that makes the action, participants, and key attributes explicit. We use

a representation of the form `PREDICATE(Role: value, Role: value, …)`. For example, `PEEL(Agent: you, Object: onion(Size: medium, Quantity: 1))`. The normalization reduces lexical variation and keeps critical task distinctions like object identity, quantity, and destination. These distinctions are especially important in our setting because many errors are expressed in step arguments and not in the main predicate. Appendix C, Section C.1 gives additional SRL details.

SRLs remove lexical variation and dataset artifacts but keep the event structure that determines procedural correctness. This is useful for CaptainCook4D, where near-duplicate steps may appear with different surface forms or different IDs, and for EgoOops, where long step descriptions are paired with short error cues. We generate SRL representations offline under a constrained prompt and a strict JSON schema, then review the outputs.

Action–object representations. We also derive an action–object abstraction that keeps only the main predicate and the salient objects involved in the action. It removes quantities, destinations, surfaces, manner, and other role fillers. We use this only as an ablation. It tests how much procedural detail can be removed before mistake detection drops.

Video-to-text grounding with VLMs. In the practical setting, the VLM receives a segmented clip and the candidate step descriptions for the corresponding task. The model selects the best matching description from this closed vocabulary and returns a structured output with at least `predicted_description`, `action_name`, and `objects`. We use `predicted_description` as the main grounded representation. The auxiliary fields are used only for the action–object ablation. If a grounded description is not covered by the cached step vocabulary, we normalize it offline into an SRL representation before evaluation.

6.6.1 Auxiliary action–object superstructure experiments

The action–object abstraction was first developed in the earlier CaptainCook4D/Assembly101 study as a lightweight way to move video closer to text when full step grounding was unreliable. We retain it here as an auxiliary representation and as a bridge to the Assembly101 experiments reported later in the chapter. Figure 6.3 illustrates the corresponding pipeline, and Appendix C reports the prompting details and auxiliary perception results for CaptainCook4D and Assembly101.

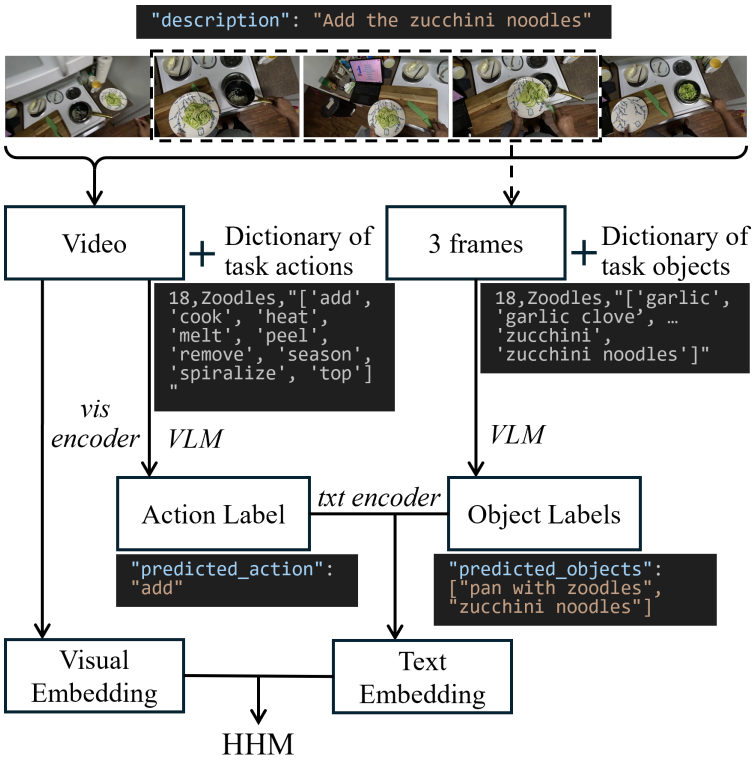


Figure 6.3: Action–object superstructure for CaptainCook4D and Assembly101. Video segments are converted into a textual trace and can be compressed into action–object descriptors before embedding and HMM inference. The experiments show that this interface can serve as a bridge from video to text, but the compressed representation is too lossy for full PCMD and works best as an auxiliary component.

6.7 Experimental Setup and Evaluation Protocol

Benchmarks and protocol. We evaluate ChronoFix under two complementary settings. The main unified-task evaluation uses CaptainCook4D [96], EgoPER [61], and EgoOops [47]. We recast all three datasets under one post-completion protocol: the input is a fully observed, step-segmented procedural video, and the model must recover the executed step sequence and detect deviations from a valid reference procedure. We follow the text augmentation and cleanup protocol from Section 6.6.

Assembly101 is retained as an auxiliary benchmark from the earlier PCMD study. It is useful for testing the limits of grounding and action–object superstructures in a visually difficult assembly domain, but it is reported separately from the main three-dataset leaderboard because its label space and protocol differ more substantially from the unified setting.

Models and representations. For visual step embeddings, we use frozen VideoMAE-v2 [128] and V-JEPA 2 [5] backbones. For text, we sweep frozen CLIP [102], all-MiniLM [130], all-MPNet [113], SimCSE [39], Instructor-XL [115], and OpenAI text-embedding-3-large. For video-to-text grounding, we compare Gemini and Qwen. On the language side, we evaluate raw step descriptions, semantic role representations, and action–object abstractions.

Metrics. We report four main metrics for the unified post-completion task. **Any-F1** is our primary metric and evaluates step-level mistake detection after collapsing all mistake types into a single positive class. **Type-aware F1** evaluates whether the model assigns the correct edit-based error type in the unified taxonomy. **Seq** measures recovery of the executed step sequence. **Step-AUC** measures threshold-free discrimination between correct and erroneous steps.

For comparison with prior work, we also report benchmark-specific metrics under the original evaluation protocols. On EgoPER, we report EDA/AUC following EgoPED and AMNAR. On CaptainCook4D, we report Precision/EDA/AUC. These benchmark-specific numbers are used only in the prior-work comparison and do not replace the unified-task metrics.

Selection rule. Each table reports one complete configuration per entry. For the unified-task tables, we report the configuration with the highest Any-F1 and all remaining metrics from that same run. For the prior-work comparison, we analogously report the configuration attaining the highest benchmark-facing metric within the explored sweep. Ground-truth dataset labels are mapped into the unified taxonomy of insertion, deletion, substitution, and transposition. When a benchmark contains additional execution-error tags, these are discussed separately and are not merged into the main leaderboard.

6.8 Results

6.8.1 Main unified-task results

Table 6.1 summarizes attainable performance by representation family. The raw-description rows use the benchmark-provided step descriptions to define a text-side upper bound. Vision-only runs are weakest in Seq on all three datasets. Grounded text closes a large part of that gap. Gemini + SRL gives the best Any-F1 on EgoOops, ties for the best Any-F1 on CaptainCook4D, and stays close on EgoPER. Among the grounded variants, raw Gemini text is best on EgoPER, because this benchmark has short atomic step descriptions, low lexical variability, and a more

	EgoOops				EgoPER				CC4D			
Family	**Seq**	**Any-F1**	**Type-F1**	**AUC**	**Seq**	**Any-F1**	**Type-F1**	**AUC**	**Seq**	**Any-F1**	**Type-F1**	**AUC**
Raw text	0.713	0.480	0.446	0.756	0.903	**0.741**	0.673	**0.918**	0.617	0.631	0.466	**0.761**
Vision	0.328	0.454	0.439	0.523	0.372	0.519	0.527	0.580	0.159	0.556	0.464	0.478
SRL	0.742	0.510	0.533	0.775	**0.913**	0.736	**0.675**	0.908	**0.638**	0.629	0.495	0.757
Qwen	0.493	0.441	0.451	0.564	0.802	0.607	0.599	0.736	0.517	0.603	0.498	0.640
Qwen+SRL	**0.821**	0.520	**0.549**	**0.791**	0.816	0.601	0.594	0.750	0.636	**0.637**	0.503	0.753
Gemini	0.557	0.408	0.361	0.610	0.845	0.642	0.589	0.748	0.577	0.603	**0.516**	0.656
Gemini+SRL	0.815	**0.521**	0.539	0.767	0.854	0.631	0.595	0.762	0.635	**0.637**	0.505	0.755

Table 6.1: Family-level leaderboard summarizing attainable performance. Each row reports the best configuration within that family, selected by Any-F1. The corresponding encoder and HMM settings differ across rows. Seq denotes sequence accuracy, and CC4D denotes CaptainCook4D. Section C.3.1 lists the corresponding configurations.

	Benchmark step descriptions			Qwen grounding			Gemini grounding		
Dataset	**Raw**	**SRL**	**Act.+Obj.**	**Raw**	**SRL**	**Act.+Obj.**	**Raw**	**SRL**	**Act.+Obj.**
EgoOops	**0.450**	0.417	0.318	0.419	**0.509**	0.444	0.389	**0.517**	0.414
EgoPER	**0.712**	0.661	0.540	0.578	**0.601**	0.550	0.602	**0.631**	0.552
CaptainCook4D	**0.605**	0.582	0.571	0.587	**0.637**	0.566	0.595	**0.637**	0.570

Table 6.2: Controlled comparison of Any-F1 across step representations. For each dataset and source block, we report one matched configuration and compare Raw, SRL, and Act.+Obj. under that setting. Section C.3.2 lists the corresponding configurations.

consistent visual environment than the other two datasets. This pattern is consistent with the fact that EgoOops and CaptainCook4D contain more complex, argument-level error signals.

6.8.2 Representations: raw text, SRL, and action–object

Table 6.2 shows the controlled representation comparison. On benchmark-provided text, Raw is strongest on all three datasets, SRL stays close, and Act.+Obj. is consistently weaker. On grounded text, SRL is strongest for both Qwen and Gemini on all three datasets, while Act.+Obj. remains below SRL throughout. This supports SRL mainly as a normalization layer for noisy VLM grounding. When benchmark text is already clean, raw descriptions remain strongest.

	Best grounded				Shared config			
Dataset	**Seq**	**Any-F1**	**Type-F1**	**AUC**	**Seq**	**Any-F1**	**Type-F1**	**AUC**
EgoOops	**0.815**	**0.521**	**0.539**	0.767	0.727	0.449	0.424	**0.775**
EgoPER	**0.845**	**0.642**	**0.589**	**0.748**	0.714	0.575	0.549	0.703
CaptainCook4D	0.635	**0.637**	**0.505**	0.755	**0.683**	0.629	0.483	**0.773**

Table 6.3: Best grounded pipeline on each dataset versus one shared grounded configuration. "Best grounded" denotes the highest-scoring grounded configuration for that dataset among the grounded families in Table 6.1, selected by Any-F1. It is given for direct comparison with the single shared configuration used across all three datasets. The shared configuration is Gemini + SRL + Instructor-XL 128-d + no normalization + cosine emission ($T = 12$) + difflib alignment.

6.8.3 One shared grounded configuration

Table 6.3 separates the best grounded pipeline on each dataset from one shared grounded configuration. The best per-dataset rows switch between raw grounded text and grounded SRLs. The shared configuration uses Gemini predicted descriptions, SRLs, Instructor-XL 128-d embeddings, no normalization, cosine emission with temperature 12, and difflib alignment. It is the strongest grounded configuration shared by all three datasets. The Any-F1 gap to the per-dataset best is small on CaptainCook4D (-0.008) and moderate on EgoPER and EgoOops. This configuration is the strongest single shared setting across all three datasets.

6.8.4 Comparison with prior work

Table 6.4 reports benchmark-specific metrics. We select the best result by EDA and report the remaining numbers from the same run. On EgoPER, all four grounded variants improve over EgoPED and AMNAR on both EDA and AUC, with raw Gemini text reaching 72.5 EDA and 76.8 AUC. On CaptainCook4D, raw grounded text is weak under the AMNAR evaluation, but SRLs close the gap sharply: Gemini + SRL reaches 80.9 Precision, 84.3 EDA, and 90.0 AUC; Qwen + SRL reaches 78.5, 84.1, and 89.9. This split between raw grounded text and grounded SRLs matches the results in Table 6.2.

6.8.5 Auxiliary Assembly101 transfer experiments

Assembly101 is retained as an auxiliary benchmark from the earlier ChronoFix study. In this domain, grounding is complicated by fine-grained part distinctions, repeated attachment/detachment cycles, and a protocol that does not align as cleanly with the later unified post-

	EgoPER		CaptainCook4D		
Method	**EDA**	**AUC**	**Precision**	**EDA**	**AUC**
EgoPED [61]	57.0	62.0	56.5	69.8	54.9
AMNAR [49]	64.4	68.5	66.8	72.3	60.2
Ours (Qwen)	70.4	75.1	25.3	64.6	69.4
Ours (Qwen + SRL)	69.9	75.3	78.5	84.1	89.9
Ours (Gemini)	**72.5**	**76.8**	29.1	71.1	75.3
Ours (Gemini + SRL)	71.3	76.2	**80.9**	**84.3**	**90.0**

Table 6.4: Comparison with prior work under benchmark-specific protocols. Each of our rows reports the highest EDA attained within that family under the corresponding benchmark protocol; the remaining metrics come from the same configuration. The methods are compared under the original benchmark metrics, but their inference pipelines differ. On CaptainCook4D, following AMNAR, the reported numbers correspond to the execution-only evaluation.

completion setting. We therefore use Assembly101 to test the limits of the HMM backbone and of lightweight action–object superstructures, not to place it on the same main leaderboard as CaptainCook4D, EgoPER, and EgoOops.

The auxiliary results support the same structural conclusion as the main experiments. Pure visual embeddings remain weak, direct step grounding is difficult, and object-centric auxiliary signals are more reliable than full step prediction. This is exactly the regime in which compressed action–object representations are useful as a bridge from video to text, even though they are still too lossy for the main language-grounded PCMD task. Appendix C reports the auxiliary perception-side comparisons: Table C.10 for CaptainCook4D, Table C.11 for Assembly101 grounding tasks, and Table C.12 for the legacy Assembly101 HMM results.

6.9 Discussion

ChronoFix is intentionally conservative in modeling: the contribution is a systems abstraction for procedural video. The earlier CaptainCook4D/Assembly101 study established that even a lightweight HMM can support interpretable PCMD once the input is expressed in step space. The later cross-benchmark study showed that the quality of that step space is decisive: raw benchmark text is the upper bound, semantic-role normalization helps most when VLM grounding is noisy, and action–object abstractions are useful as a lightweight bridge but too lossy for the main task. Together, these results support a modular pipeline in which stronger grounding can be plugged into the same explicit procedure model.

6.10 Summary

ChronoFix demonstrates that PCMD becomes much more tractable when perception and verification are decomposed. Across CaptainCook4D, EgoPER, and EgoOops, language-grounded HMM inference provides a compact and interpretable framework for sequence-level mistake detection, and semantic-role normalization improves robustness when grounded text is noisy. Auxiliary Assembly101 experiments confirm the same structural point in a harder assembly domain and clarify the limit of compressed action–object representations.

For PCMD, the most reliable route is to separate VLM-based perception from explicit probabilistic task structure.

This path is, in our view, a productive route to a portable foundation for real-world procedural quality control and transparent human–AI collaboration.

Chapter 7

Conclusions and Future Work

This thesis studied Post-Completion Mistake Detection (PCMD) in procedural videos: the problem of verifying a completed multi-step execution against an intended protocol, localizing deviations, and characterizing their type. PCMD remains under-explored compared to online settings, yet it is central for real-world verification scenarios where the full execution is available and correctness must be assessed reliably. The thesis approached PCMD as a language-centered verification problem grounded in cognitive insights, temporal reasoning diagnostics, and structured sequence modeling.

7.1 Summary of contributions

The thesis develops a unified framework for PCMD through three complementary contributions.

Diagnostics of VLM limitations. Chapters 3 and 4 provided diagnostic evidence that current end-to-end VLM reasoning is unreliable for procedural verification prerequisites. The analyses highlighted systematic failures in temporal reasoning (in particular, confusions around completion, ordering, and overlap) and showed that evaluation formats such as multiple-choice VideoQA can obscure these failures through selection artifacts and "blind guessing" behavior.

Mistake-aware data and dataset assessment (PIE-V). Chapter 5 addressed the data bottleneck by introducing a unified view of procedural mistakes and corrections, together with an assessment rubric for comparing mistake-aware datasets in terms of plausibility, coherence, and grounding. It also described PIE-V, a plan-first pipeline for injecting controlled,

semantics-aware mistakes and plausible corrections into clean procedures, supporting scalable dataset enrichment and reproducible robustness evaluation.

Language-grounded, structure-first PCMD modeling (ChronoFix). Chapter 6 proposed an efficient and interpretable baseline for PCMD: a language-grounded Hidden Markov Model trained on correct-only step traces, combined with alignment and edit-based diagnosis under a unified taxonomy. The model shifts the core of mistake detection into step-sequence space and produces predictions that can be traced to concrete transitions, likelihood signals, and edit operations.

7.2 What the thesis changes for PCMD practice

This thesis changes PCMD practice in three ways.

First, it reframes PCMD as *verification under temporal–causal constraints*, rather than as a direct extension of action recognition or generic VideoQA. The diagnostic studies clarify which sub-skills are non-negotiable for verification (completion-sensitive reasoning, correct temporal binding, protocol consistency) and how common evaluation setups can misrepresent model competence.

Second, it treats *mistakes and corrections* as core modeling and dataset objects. Instead of focusing only on error labels, the thesis emphasizes realism, logical consistency, and text–video grounding quality, and provides a rubric and a pipeline that make these properties measurable and improvable.

Third, it offers a practical route to interpretable systems. By operating in text space over grounded step sequences, PCMD decisions become explainable and easier to calibrate. This reduces reliance on opaque end-to-end reasoning and supports resource-conscious deployment, transfer across domains, and more controlled error analysis.

7.3 Future work

7.3.1 Richer language explanations of detected mistakes

The current ChronoFix framework already uses semantic-role-aware structure as an intermediate representation for robust mistake detection. A natural next step is therefore to build an explanation layer on top of this representation: given a detected deviation, the system should verbalize which step was expected, what was observed instead, and which role mismatch or se-

quencing violation triggered the diagnosis. This would turn the current structured error signal into user-facing feedback for tutoring, troubleshooting, and human-in-the-loop verification.

7.3.2 Better grounding under open VLMs

Grounding quality remains a key bottleneck: if a grounded step trace is unstable, downstream detection cannot be reliable. Future work should explore stronger open grounding pipelines, including training-free step grounding methods, better temporal localization, and calibration-aware confidence measures for step hypotheses. A related direction is to model grounding uncertainty explicitly, for example through step lattices rather than a single best sequence.

7.3.3 From post-completion to online/early settings

While this thesis focused on PCMD, the structural view extends to online and early detection. A promising direction is to adapt the sequence-based framework to partial traces, where the model maintains a belief over the current step and predicts likely continuations, allowing it to flag deviations before they fully unfold. This requires careful treatment of prefix ambiguity and of the trade-off between early warning and false alarms.

7.3.4 Dataset releases and evaluation standardization

Finally, progress in mistake detection depends on shared evaluation standards. Future work includes releasing mistake-enriched data generated with PIE-V, expanding audits across domains, and standardizing dataset reporting along dimensions captured by the rubric (realism, coherence, grounding, corrections). Establishing common taxonomies and comparable evaluation protocols would make results more transferable and reduce dataset-specific overfitting.

7.3.5 From PIE-V-style generation to model-ready benchmarks

An important next step is to connect the data and modeling contributions more directly. PIE-V validates controllable, psychologically plausible mistake injection across heterogeneous procedures. ChronoFix needs repeated correct executions per canonical task so that sequence statistics can be estimated reliably. A future benchmark should combine both properties: multiple correct executions per task together with mistake-enriched variants derived from them. This would provide a principled setting for training and evaluating structure-first models on realistic mistake traces.

Overall, this thesis argues for a structured, language-centered approach to procedural verification: diagnose the limits of black-box reasoning, build realistic mistake-aware data, and detect deviations through interpretable sequence modeling. This combination provides a foundation for more reliable procedural assistants and for a more mature research ecosystem around PCMD.

Bibliography

[1] Marah Abdin, Jyoti Aneja, Hany Awadalla, Ahmed Awadallah, Ammar Ahmad Awan, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Jianmin Bao, Harkirat Behl, Alon Benhaim, Misha Bilenko, Johan Bjorck, Sébastien Bubeck, Martin Cai, Qin Cai, Vishrav Chaudhary, Dong Chen, Dongdong Chen, Weizhu Chen, Yen-Chun Chen, Yi-Ling Chen, Hao Cheng, Parul Chopra, Xiyang Dai, Matthew Dixon, Ronen Eldan, Victor Fragoso, Jianfeng Gao, Mei Gao, Min Gao, Amit Garg, Allie Del Giorno, Abhishek Goswami, Suriya Gunasekar, Emman Haider, Junheng Hao, Russell J. Hewett, Wenxiang Hu, Jamie Huynh, Dan Iter, Sam Ade Jacobs, Mojan Javaheripi, Xin Jin, Nikos Karampatziakis, Piero Kauffmann, Mahoud Khademi, Dongwoo Kim, Young Jin Kim, Lev Kurilenko, James R. Lee, Yin Tat Lee, Yuanzhi Li, Yunsheng Li, Chen Liang, Lars Liden, Xihui Lin, Zeqi Lin, Ce Liu, Liyuan Liu, Mengchen Liu, Weishung Liu, Xiaodong Liu, Chong Luo, Piyush Madan, Ali Mahmoudzadeh, David Majercak, Matt Mazzola, Caio César Teodoro Mendes, Arindam Mitra, Hardik Modi, Anh Nguyen, Brandon Norick, Barun Patra, Daniel Perez-Becker, Thomas Portet, Reid Pryzant, Heyang Qin, Marko Radmilac, Liliang Ren, Gustavo de Rosa, Corby Rosset, Sambudha Roy, Olatunji Ruwase, Olli Saarikivi, Amin Saied, Adil Salim, Michael Santacroce, Shital Shah, Ning Shang, Hiteshi Sharma, Yelong Shen, Swadheen Shukla, Xia Song, Masahiro Tanaka, Andrea Tupini, Praneetha Vaddamanu, Chunyu Wang, Guanhua Wang, Lijuan Wang, Shuohang Wang, Xin Wang, Yu Wang, Rachel Ward, Wen Wen, Philipp Witte, Haiping Wu, Xiaoxia Wu, Michael Wyatt, Bin Xiao, Can Xu, Jiahang Xu, Weijian Xu, Jilong Xue, Sonali Yadav, Fan Yang, Jianwei Yang, Yifan Yang, Ziyi Yang, Donghan Yu, Lu Yuan, Chenruidong Zhang, Cyril Zhang, Jianwen Zhang, Li Lyna Zhang, Yi Zhang, Yue Zhang, Yunan Zhang, and Xiren Zhou. Phi-3 technical report: A highly capable language model locally on your phone, 2024.

[2] Mary Abkemeier. Cognitive load theory. *Encyclopedia of Education and Information Technologies*, 2020.

[3] James F. Allen. Towards a general theory of action and time. *Artificial Intelligence*, 23

(2):123–154, 1984.

[4] Jacopo Amidei, Paul Piwek, and Alistair Willis. The use of rating and Likert scales in natural language generation human evaluation tasks: A review and some recommendations. In Kees van Deemter, Chenghua Lin, and Hiroya Takamura, editors, *Proceedings of the 12th International Conference on Natural Language Generation*, pages 397–402, Tokyo, Japan, October−November 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-8648. URL https://aclanthology.org/W19-8648/.

[5] Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Mojtaba, Komeili, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, Sergio Arnaud, Abha Gejji, Ada Martin, Francois Robert Hogan, Daniel Dugas, Piotr Bojanowski, Vasil Khalidov, Patrick Labatut, Francisco Massa, Marc Szafraniec, Kapil Krishnakumar, Yong Li, Xiaodong Ma, Sarath Chandar, Franziska Meier, Yann LeCun, Michael Rabbat, and Nicolas Ballas. V-JEPA 2: Self-supervised video models enable understanding, prediction and planning, 2025. URL https://arxiv.org/abs/2506.09985.

[6] Konstantinos Bacharidis and Antonis A. Argyros. Vision-based mistake analysis in procedural activities: A review of advances and challenges, 2025.

[7] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[8] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. Frozen in time: A joint video and image encoder for end-to-end retrieval. In *IEEE International Conference on Computer Vision*, 2021.

[9] Mohamad Azhari Abu Bakar, Kartini Abd Ghani, and Norehan Zulkiply. Reviewing how individual differences in working memory capacity affect the ability in following instructions. *Malaysian Journal of Social Sciences and Humanities (MJSSH)*, 2023.

[10] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. The Berkeley FrameNet project. In *36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Volume 1*, pages 86–90, Montreal, Quebec, Canada, August 1998. Association for Computational Linguistics. doi: 10.3115/980845.980860. URL https://aclanthology.org/P98-1013/.

[11] Siddhant Bansal, Chetan Arora, and C. V. Jawahar. My view is the best view: Procedure learning from egocentric videos, 2022.

[12] Yizhak Ben-Shabat, Xin Yu, Fatemeh Sadat Saleh, Dylan Campbell, Cristian Rodriguez-Opazo, Hongdong Li, and Stephen Gould. The ikea asm dataset: Understanding people assembling furniture through actions, objects and pose, 2023.

[13] Jasmijn E. Bosch, Mathilde Chailleux, and Francesca Foppolo. Incremental processing of telicity in italian children. *Experiments in Linguistic Meaning*, 1:15–26, Jul 2021. doi: 10.3765/elm.1.4880.

[14] Alexandra Bremers, Alexandria Pabst, Maria Teresa Parreira, and Wendy Ju. Using social cues to recognize task failures for hri: Overview, state-of-the-art, and future directions, 2024.

[15] Michael D. Byrne and Susan Bovair. A working memory model of a common procedural error. *Cogn. Sci.*, 21:31–61, 1997.

[16] Michael D. Byrne and Elizabeth M. Davis. Task structure and postcompletion error in the execution of a routine procedure. *Human Factors*, 48(4):627–638, 2006. doi: 10.1518/001872006779166398. URL https://doi.org/10.1518/001872006779166398.

[17] Mu Cai, Reuben Tan, Jianrui Zhang, Bocheng Zou, Kai Zhang, Feng Yao, Fangrui Zhu, Jing Gu, Yiwu Zhong, Yuzhang Shang, Yao Dou, Jaden Park, Jianfeng Gao, Yong Jae Lee, and Jianwei Yang. Temporalbench: Towards fine-grained temporal understanding for multimodal video models. *arXiv preprint arXiv:2410.10818*, 2024.

[18] Dominic Callan and Jennifer Foster. How interesting and coherent are the stories generated by a large-scale neural language model? comparing human and automatic evaluations of machine-generated text. *Expert Systems*, 40, 2023.

[19] Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. INSIDE: LLMs' internal states retain the power of hallucination detection. In *The Twelfth International Conference on Learning Representations*, 2024.

[20] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. *arXiv preprint arXiv:2312.14238*, 2023.

[21] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL https://lmsys.org/blog/2023-03-30-vicuna/.

[22] Hyung Won Chung, Le Hou, S. Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Wei Yu, Vincent Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed Huai hsin Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *ArXiv*, abs/2210.11416, 2022.

[23] Daniel Cores, Michael Dorkenwald, Manuel Mucientes, Cees G. M. Snoek, and Yuki M. Asano. Lost in time: A new temporal benchmark for videollms. In *European Conference on Computer Vision (ECCV)*, 2024.

[24] Gautier Dagan, Olga Loginova, and Anil Batra. CAST: Cross-modal alignment similarity test for vision language models. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 1387–1402, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.93/.

[25] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. The epic-kitchens dataset: Collection, challenges and baselines, 2020.

[26] Guodong Ding, Fadime Sener, Shugao Ma, and Angela Yao. Every mistake counts in assembly. *ArXiv*, abs/2307.16453, 2023.

[27] Guodong Ding, Fadime Sener, and Angela Yao. Temporal action segmentation: An analysis of modern techniques, 2023.

[28] Xi Ding and Lei Wang. The journey of action recognition. *Companion Proceedings of the ACM on Web Conference 2025*, 2025.

[29] Sabrina Dunham, Edward Lee, and Adam M. Persky. The psychology of following instructions and its implications. *American Journal of Pharmaceutical Education*, 84, 2020.

[30] Lisa Dunlap, Yuhui Zhang, Xiaohan Wang, Ruiqi Zhong, Trevor Darrell, Jacob Steinhardt, Joseph E. Gonzalez, and Serena Yeung-Levy. Describing differences in image sets with natural language. *ArXiv*, abs/2312.02974, 2023.

[31] Nicholas D. Duran, Philip M. McCarthy, Arthur C. Graesser, and Danielle S. McNamara. Using temporal cohesion to predict temporal coherence in narrative and expository texts. *Behavior Research Methods*, 39(2):212–223, 2007. doi: 10.3758/BF03193150.

[32] Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031, 2021. doi: 10.1162/tacl_a_00410. URL https://aclanthology.org/2021.tacl-1.60.

[33] Jeffrey L. Elman and Ken McRae. A model of event knowledge. *Psychological Review*, 126:252–291, 2019.

[34] Alessandro Flaborea, Guido Maria D'Amely di Melendugno, Leonardo Plini, Luca Scofano, Edoardo De Matteis, Antonino Furnari, G. Farinella, and Fabio Galasso. Prego: Online mistake detection in procedural egocentric videos. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18483–18492, 2024.

[35] Francesca Foppolo, Francesca Panzeri, Cesare Greco, and Matteo Carminati. The incremental processing of accomplishment predicates. In *Proceedings of the 38th Annual Conference of the Cognitive Science Society (CogSci)*, pages 2572–2577, 2016.

[36] Francesca Foppolo, Jasmijn E. Bosch, Ciro Greco, Maria Nella Carminati, and Francesca Panzeri. Draw a star and make it perfect: Incremental processing of telicity. *Cognitive science*, 45 10:e13052, 2021.

[37] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Rongrong Ji, and

Xing Sun. Video-mme: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis, 2024.

[38] Xingyu Fu, Yushi Hu, Bangzheng Li, Yu Feng, Haoyu Wang, Xudong Lin, Dan Roth, Noah A. Smith, Wei-Chiu Ma, and Ranjay Krishna. Blink: Multimodal large language models can see but not perceive. *ArXiv*, abs/2404.12390, 2024.

[39] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings, 2022.

[40] Sara Garfield and B.D. Franklin. Understanding models of error in clinical practice. *Pharmaceutical Journal*, 296:361–364, 06 2016.

[41] Reza Ghoddoosian, Isht Dwivedi, Nakul Agarwal, and Behzad Dariush. Weakly-supervised action segmentation and unseen error detection in anomalous instructional videos. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10094–10104, 2023.

[42] Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002. doi: 10.1162/089120102760275983. URL https://aclanthology.org/J02-3001/.

[43] Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind: One embedding space to bind them all. In *CVPR*, 2023.

[44] Kristen Grauman, Andrew Westbury, Lorenzo Torresani, Kris Kitani, Jitendra Malik, Triantafyllos Afouras, Kumar Ashutosh, Vijay Baiyya, Siddhant Bansal, Bikram Boote, Eugene Byrne, Zach Chavis, Joya Chen, Feng Cheng, Fu-Jen Chu, Sean Crane, Avijit Dasgupta, Jing Dong, Maria Escobar, Cristhian Forigua, Abrham Gebreselasie, Sanjay Haresh, Jing Huang, Md Mohaiminul Islam, Suyog Jain, Rawal Khirodkar, Devansh Kukreja, Kevin J Liang, Jia-Wei Liu, Sagnik Majumder, Yongsen Mao, Miguel Martin, Effrosyni Mavroudi, Tushar Nagarajan, Francesco Ragusa, Santhosh Kumar Ramakrishnan, Luigi Seminara, Arjun Somayazulu, Yale Song, Shan Su, Zihui Xue, Edward Zhang, Jinxu Zhang, Angela Castillo, Changan Chen, Xinzhu Fu, Ryosuke Furuta, Cristina Gonzalez, Prince Gupta, Jiabo Hu, Yifei Huang, Yiming Huang, Weslie Khoo, Anush Kumar, Robert Kuo, Sach Lakhavani, Miao Liu, Mi Luo, Zhengyi Luo, Brighid Meredith, Austin Miller, Oluwatumininu Oguntola, Xiaqing Pan, Penny Peng, Shraman Pramanick, Merey Ramazanova, Fiona Ryan, Wei Shan, Kiran Somasundaram, Chenan

Song, Audrey Southerland, Masatoshi Tateno, Huiyu Wang, Yuchen Wang, Takuma Yagi, Mingfei Yan, Xitong Yang, Zecheng Yu, Shengxin Cindy Zha, Chen Zhao, Ziwei Zhao, Zhifan Zhu, Jeff Zhuo, Pablo Arbelaez, Gedas Bertasius, David Crandall, Dima Damen, Jakob Engel, Giovanni Maria Farinella, Antonino Furnari, Bernard Ghanem, Judy Hoffman, C. V. Jawahar, Richard Newcombe, Hyun Soo Park, James M. Rehg, Yoichi Sato, Manolis Savva, Jianbo Shi, Mike Zheng Shou, and Michael Wray. Ego-exo4d: Understanding skilled human activity from first- and third-person perspectives, 2024.

[45] Kaisi Guan, Zhengfeng Lai, Yuchong Sun, Peng Zhang, Wei Liu, Kieran Liu, Meng Cao, and Ruihua Song. Etva: Evaluation of text-to-video alignment via fine-grained question generation and answering, 2025.

[46] Wenliang Guo, Yujiang Pu, and Yu Kong. Procedural mistake detection via action effect modeling, 2025.

[47] Yuto Haneji, Taichi Nishimura, Hirotaka Kameko, Keisuke Shirai, Tomoya Yoshida, Keiya Kajimura, Koki Yamamoto, Taiyu Cui, Tomohiro Nishimoto, and Shinsuke Mori. Egooops: A dataset for mistake action detection from egocentric videos referring to procedural texts, 2025.

[48] Muyang He, Yexin Liu, Boya Wu, Jianhao Yuan, Yueze Wang, Tiejun Huang, and Bo Zhao. Efficient multimodal learning from data-centric perspective. *arXiv preprint arXiv:2402.11530*, 2024.

[49] Wei-Jin Huang, Yuan-Ming Li, Zhi-Wei Xia, Yu-Ming Tang, Kun-Yu Lin, Jian-Fang Hu, and Wei-Shi Zheng. Modeling multiple normal action representations for error detection in procedural tasks, 2025.

[50] Youngkyoon Jang, Brian T. Sullivan, Casimir J. H. Ludwig, Iain D. Gilchrist, Dima Damen, and W. Mayol-Cuevas. Epic-tent: An egocentric video dataset for camping tent assembly. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 4461–4469, 2019.

[51] Yunseok Jang, Yale Song, Chris Dongjoo Kim, Youngjae Yu, Youngjin Kim, and Gunhee Kim. Video question answering with spatio-temporal reasoning. *Int. J. Comput. Vision*, 127(10):1385–1412, October 2019. ISSN 0920-5691. doi: 10.1007/s11263-019-01189-x. URL https://doi.org/10.1007/s11263-019-01189-x.

[52] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[53] Marcel Adam Just and Patricia A. Carpenter. A capacity theory of comprehension: individual differences in working memory. *Psychological review*, 99 1:122–49, 1992.

[54] Kushal Kafle and Christopher Kanan. Visual question answering: Datasets, algorithms, and future challenges. *Computer Vision and Image Understanding*, 163:3–20, October 2017. ISSN 1077-3142. doi: 10.1016/j.cviu.2017.06.005. URL http://dx.doi.org/10.1016/j.cviu.2017.06.005.

[55] Hilde Kuehne, Ali Bilgin Arslan, and Thomas Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 780–787, 2014.

[56] Chi-Hsi Kung, Frangil Ramirez, Juhyung Ha, Yi-Ting Chen, David Crandall, and Yi-Hsuan Tsai. What changed and what could have changed? state-change counterfactuals for procedure-aware video representation learning, 2025.

[57] Bolin Lai, Sam Toyer, Tushar Nagarajan, Rohit Girdhar, Shengxin Zha, James M. Rehg, Kris Kitani, Kristen Grauman, Ruta Desai, and Miao Liu. Human action anticipation: A survey, 2024.

[58] Hugo Laurençon, Andrés Marafioti, Victor Sanh, and Léo Tronchon. Building and better understanding vision-language models: insights and future directions., 2024.

[59] {Chris van der} Lee, Albert Gatt, {Emiel van} Miltenburg, and Emiel Krahmer. Human evaluation of automatically generated text: Current trends and best practice guidelines. *Computer Speech and Language: An official publication of the International Speech Communication Association (ISCA)*, 67:1–24, May 2021. ISSN 0885-2308. doi: 10.1016/j.csl.2020.101151. Funding Information: We received support from RAAK-PRO SIA (2014-01-51PRO) and The Netherlands Organization for Scientific Research (NWO 360-89-050), which is gratefully acknowledged. Furthermore, we want to extend our gratitude towards the anonymous reviewers and also towards Leshem Choshen, Ond?ej Du?ek, Kees van Deemter, Dimitra Gkatzia, David Howcroft, Ehud Reiter, and Sander Wubben for their valuable comments on the paper. Funding Information: We received support from RAAK-PRO SIA (2014-01-51PRO) and The Netherlands Organization for Scientific Research (NWO 360-89-050), which is gratefully acknowledged.

Furthermore, we want to extend our gratitude towards the anonymous reviewers and also towards Leshem Choshen, Ondřej Dušek, Kees van Deemter, Dimitra Gkatzia, David Howcroft, Ehud Reiter, and Sander Wubben for their valuable comments on the paper. Publisher Copyright: © 2020 The Authors.

[60] Shih-Po Lee and Ehsan Elhamifar. Error recognition in procedural videos using generalized task graph. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10009–10021, October 2025.

[61] Shih-Po Lee, Zijia Lu, Zekun Zhang, Minh Hoai, and Ehsan Elhamifar. Error detection in egocentric procedural task videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 18655–18666, June 2024.

[62] Dan Lehman, Tim J. Schoonbeek, Shao-Hsuan Hung, Jacek Kustra, Peter H. N. de With, and Fons van der Sommen. Find the assembly mistakes: Error segmentation for industrial applications, 2024.

[63] Junlong Li, Huaiyuan Xu, Sijie Cheng, Kejun Wu, Kim-Hui Yap, Lap-Pui Chau, and Yi Wang. Building egocentric procedural ai assistant: Methods, benchmarks, and challenges, 2025.

[64] Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *ArXiv*, abs/2301.12597, 2023.

[65] Qing Li, Chenyang Lyu, Jiahui Geng, Derui Zhu, Maxim Panov, and Fakhri Karray. Reference-free hallucination detection for large vision-language models. *arXiv preprint arXiv:2408.06733*, 2024.

[66] Wangyue Li, Liangzhi Li, Tong Xiang, Xiao Liu, Wei Deng, and Noa Garcia. Can multiple-choice questions really be useful in detecting the abilities of llms? *ArXiv*, abs/2403.17752, 2024.

[67] Yayuan Li, Aadit Jain, Filippos Bellos, and Jason J. Corso. Mistake attribution: Fine-grained mistake understanding in egocentric videos, 2025.

[68] Mingxiang Liao, Hannan Lu, Xinyu Zhang, Fang Wan, Tianyu Wang, Yuzhong Zhao, Wangmeng Zuo, Qixiang Ye, and Jingdong Wang. Evaluation of text-to-video generation models: A dynamics perspective, 2024.

[69] Bin Lin, Bin Zhu, Yang Ye, Munan Ning, Peng Jin, and Li Yuan. Video-llava: Learning united visual representation by alignment before projection. *ArXiv*, abs/2311.10122, 2023.

[70] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[71] Yuanxin Liu, Shicheng Li, Yi Liu, Yuxiang Wang, Shuhuai Ren, Lei Li, Sishuo Chen, Xu Sun, and Lu Hou. Tempcompass: Do video llms really understand videos?, 2024.

[72] Yuanzhan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. Mmbench: Is your multi-modal model an all-around player? *ArXiv*, abs/2307.06281, 2023.

[73] Olga Loginova and Raffaella Bernardi. The inherence of telicity: Unveiling temporal reasoning in video question answering. In Federico Boschetti, Gianluca E. Lebani, Bernardo Magnini, and Nicole Novielli, editors, *Proceedings of the Ninth Italian Conference on Computational Linguistics (CLiC-it 2023)*, pages 546–550, Venice, Italy, November 2023. CEUR Workshop Proceedings. ISBN 979-12-550-0084-6. URL https://aclanthology.org/2023.clicit-1.72/.

[74] Olga Loginova and Frank Keller. How to correctly make mistakes: A framework for constructing and benchmarking mistake aware egocentric procedural videos, 2026. URL https://arxiv.org/abs/2604.15134.

[75] Olga Loginova and Sofía Ortega Loguinova. Deep temporal reasoning in video language models: A cross-linguistic evaluation of action duration and completion through perfect times. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 20472–20502, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.1000. URL https://aclanthology.org/2025.acl-long.1000/.

[76] Olga Loginova, Oleksandr Bezrukov, Ravi Shekhar, and Alexey Kravets. Addressing blind guessing: Calibration of selection bias in multiple-choice question answering by video language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3216–3246,

Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.162. URL https://aclanthology.org/2025.acl-long.162/.

[77] Neil A. Macmillan and C. Douglas Creelman. *Detection Theory: A User's Guide*. Cambridge University Press, 1991.

[78] Potsawee Manakul, Adian Liusie, and Mark Gales. SelfCheckGPT: Zero-resource black-box hallucination detection for generative large language models. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9004–9017, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.557. URL https://aclanthology.org/2023.emnlp-main.557.

[79] Michele Mazzamuto, Antonino Furnari, Yoichi Sato, and Giovanni Maria Farinella. Gazing into missteps: Leveraging eye-gaze for unsupervised mistake detection in egocentric videos of skilled human activities. *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8310–8320, 2024.

[80] Stephen McKenna and Sebastian Stein. 50 salads. https://discovery.dundee.ac.uk/en/datasets/50-salads/, 2012.

[81] Eleni Metheniti, Tim Van De Cruys, and Nabil Hathout. About time: Do transformers learn temporal verbal aspect? In Emmanuele Chersoni, Nora Hollenstein, Cassandra Jacobs, Yohei Oseki, Laurent Prévot, and Enrico Santus, editors, *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics*, pages 88–101, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.cmcl-1.10. URL https://aclanthology.org/2022.cmcl-1.10/.

[82] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips, 2019.

[83] Serge Minor, Natalia Mitrofanova, Gustavo Guajardo, Myrte Vos, and Gillian Catriona Ramchand. Temporal information and event bounding across languages: Evidence from visual world eyetracking from. *Semantics and Linguistic Theory*, 2022.

[84] Marc Moens and Mark Steedman. Temporal ontology and temporal reference. *Comput. Linguist.*, 14(2):15–28, jun 1988. ISSN 0891-2017.

[85] Kosuke Moriwaki, Gaku Nakano, and Tetsuo Inoshita. The brio-ta dataset: Understanding anomalous assembly process in manufacturing. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 1991–1995, 2022. doi: 10.1109/ICIP46576.2022.9897369.

[86] Niels Mündler, Jingxuan He, Slobodan Jenko, and Martin Vechev. Self-contradictory hallucinations of large language models: Evaluation, detection and mitigation. In *The Twelfth International Conference on Learning Representations*, 2024.

[87] Medhini Narasimhan, Licheng Yu, Sean Bell, Ning Zhang, and Trevor Darrell. Learning and verification of task structure in instructional videos, 2023.

[88] Donald A. Norman. *The Psychology of Everyday Things*. Basic Books, New York, 1988. ISBN 0-465-06709-3.

[89] Donald A. Norman. *The Design of Everyday Things*. Basic Books, New York, N.Y., with a new introduction by the author edition, 2002. ISBN 0465067107.

[90] Yasumasa Onoe, Sunayana Rane, Zachary Berger, Yonatan Bitton, Jaemin Cho, Roopal Garg, Alexander Ku, Zarana Parekh, Jordi Pont-Tuset, Garrett Tanzer, et al. Docci: Descriptions of connected and contrasting images. *arXiv preprint arXiv:2404.19753*, 2024.

[91] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris,

Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-

4 technical report, 2024.

[92] Fred Paas, Alexander Renkl, and John Sweller. Cognitive load theory and instructional design: Recent developments. *Educational Psychologist*, 38:1 – 4, 2003.

[93] Martha Palmer, Daniel Gildea, and Paul Kingsbury. The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106, 2005. doi: 10.1162/0891201053630264. URL https://aclanthology.org/J05-1004/.

[94] Letitia Parcalabescu and Anette Frank. Do vision & language decoders use images and text equally? how self-consistent are their explanations? *ArXiv*, abs/2404.18624, 2024.

[95] Terence Parsons. *Events in the Semantics of English: A Study in Subatomic Semantics*. MIT Press, 1990.

[96] Rohith Peddi, Shivvrat Arya, Bharath Challa, Likhitha Pallapothula, Akshay Vyas, Bhavya Gouripeddi, Jikai Wang, Qifan Zhang, Vasundhara Komaragiri, Eric Ragan, Nicholas Ruozzi, Yu Xiang, and Vibhav Gogate. Captaincook4d: A dataset for understanding errors in procedural activities, 2024.

[97] Charles Perrow. *Normal Accidents: Living with High-Risk Technologies*. Basic Books, New York, 1984.

[98] Pouya Pezeshkpour and Estevam Hruschka. Large language models sensitivity to the order of options in multiple-choice questions. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2006–2017, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.130. URL https://aclanthology.org/2024.findings-naacl.130.

[99] Leonardo Plini, Luca Scofano, Edoardo De Matteis, Guido Maria D'Amely di Melendugno, Alessandro Flaborea, Andrea Sanchietti, G. Farinella, Fabio Galasso, and Antonino Furnari. Ti-prego: Chain of thought and in-context learning for online mistake detection in procedural egocentric videos. *ArXiv*, abs/2411.02570, 2024.

[100] Viorica Pătrăucean, Lucas Smaira, Ankush Gupta, Adrià Recasens Continente, Larisa Markeeva, Dylan Banarse, Skanda Koppula, Joseph Heyward, Mateusz Malinowski, Yi Yang, Carl Doersch, Tatiana Matejovicova, Yury Sulsky, Antoine Miech, Alex Frechette, Hanna Klimczak, Raphael Koster, Junlin Zhang, Stephanie Winkler, Yusuf

Aytar, Simon Osindero, Dima Damen, Andrew Zisserman, and João Carreira. Perception test: A diagnostic benchmark for multimodal video models. In *Advances in Neural Information Processing Systems*, 2023.

[101] Yicheng Qian, Weixin Luo, Dongze Lian, Xu Tang, Peilin Zhao, and Shenghua Gao. Svip: Sequence verification for procedures in videos, 2022.

[102] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021.

[103] Francesco Ragusa, Antonino Furnari, Salvatore Livatino, and Giovanni Maria Farinella. The meccano dataset: Understanding human-object interactions from egocentric videos in an industrial-like domain, 2020.

[104] Marvin Rausand and Arnljot Høyland. *System Reliability Theory: Models, Statistical Methods and Applications*. Wiley-Interscience, Hoboken, NJ, 2004.

[105] James Reason. *Human Error*. Cambridge University Press, 1990.

[106] James Reason. Human error: models and management. *BMJ*, 320 7237:768–70, 2000.

[107] Tim J. Schoonbeek, Tim Houben, Hans Onvlee, Peter H. N. de With, and Fons van der Sommen. Industreal: A dataset for procedure step recognition handling execution errors in egocentric videos in an industrial-like setting, 2023.

[108] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 815−823. IEEE, June 2015. doi: 10.1109/cvpr.2015.7298682. URL http://dx.doi.org/10.1109/CVPR.2015.7298682.

[109] Melanie Sclar, Yejin Choi, Yulia Tsvetkov, and Alane Suhr. Quantifying language models' sensitivity to spurious features in prompt design or: How i learned to start worrying about prompt formatting. In *The Twelfth International Conference on Learning Representations*, 2024.

[110] Luigi Seminara, Giovanni Maria Farinella, and Antonino Furnari. Differentiable task graph learning: Procedural activity representation and online mistake detection from egocentric videos, 2025.

[111] Fadime Sener, Dibyadip Chatterjee, Daniel Shelepov, Kun He, Dipika Singhania, Robert Wang, and Angela Yao. Assembly101: A large-scale multi-view video dataset for understanding procedural activities. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 21064–21074, 2022. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9878834`.

[112] M. Smits, P.P. Groenewegen, D.R.M. Timmermans, G. {van der Wal}, and C. Wagner. The nature and causes of unintended events reported at ten emergency departments. *BMC Emergency Medicine [E]*, 9:16, 2009. ISSN 1471-227X.

[113] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding, 2020.

[114] Shane Storks, Itamar Bar-Yossef, Yayuan Li, Zheyuan Zhang, Jason J. Corso, and Joyce Chai. Transparent and coherent procedural mistake detection, 2025.

[115] Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings, 2023.

[116] Franklin P. Tamborello and J. Gregory Trafton. A long-term memory competitive process model of a common procedural error. *Cognitive Science*, 35, 2013.

[117] Yansong Tang, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. Coin: A large-scale dataset for comprehensive instructional video analysis, 2019.

[118] Yolo Y. Tang, Jing Bi, Siting Xu, Luchuan Song, Susan Liang, Teng Wang, Daoan Zhang, Jie An, Jingyang Lin, Rongyi Zhu, Ali Vosoughi, Chao Huang, Zeliang Zhang, Pinxin Liu, Mingqian Feng, Feng Zheng, Jianguo Zhang, Ping Luo, Jiebo Luo, and Chenliang Xu. Video understanding with large language models: A survey, 2025.

[119] Zhengxu Tang, Zizheng Wang, Luning Wang, Zitao Shuai, Chenhao Zhang, Siyu Qian, Yirui Wu, Bohao Wang, Haosong Rao, Zhenyu Yang, and Chenwei Wu. Seqbench: Benchmarking sequential narrative generation in text-to-video models, 2025.

[120] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy,

Michael Isard, Paul R. Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Jack Krawczyk, Cosmo Du, Ed Chi, Heng-Tze Cheng, Eric Ni, Purvi Shah, Patrick Kane, Betty Chan, Manaal Faruqui, Aliaksei Severyn, Hanzhao Lin, YaGuang Li, Yong Cheng, Abe Ittycheriah, Mahdis Mahdieh, Mia Chen, Pei Sun, Dustin Tran, Sumit Bagri, Balaji Lakshminarayanan, Jeremiah Liu, Andras Orban, Fabian Güra, Hao Zhou, Xinying Song, Aurelien Boffy, Harish Ganapathy, Steven Zheng, HyunJeong Choe, Ágoston Weisz, Tao Zhu, Yifeng Lu, Siddharth Gopal, Jarrod Kahn, Maciej Kula, Jeff Pitman, Rushin Shah, Emanuel Taropa, Majd Al Merey, Martin Baeuml, Zhifeng Chen, Laurent El Shafey, Yujing Zhang, Olcan Sercinoglu, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, Alexandre Frechette, Charlotte Smith, Laura Culp, Lev Proleev, Yi Luan, Xi Chen, James Lottes, Nathan Schucher, Federico Lebron, Alban Rrustemi, Natalie Clay, Phil Crone, Tomas Kocisky, Jeffrey Zhao, Bartek Perz, Dian Yu, Heidi Howard, Adam Bloniarz, Jack W. Rae, Han Lu, Laurent Sifre, Marcello Maggioni, Fred Alcober, Dan Garrette, Megan Barnes, Shantanu Thakoor, Jacob Austin, Gabriel Barth-Maron, William Wong, Rishabh Joshi, Rahma Chaabouni, Deeni Fatiha, Arun Ahuja, Gaurav Singh Tomar, Evan Senter, Martin Chadwick, Ilya Kornakov, Nithya Attaluri, Iñaki Iturrate, Ruibo Liu, Yunxuan Li, Sarah Cogan, Jeremy Chen, Chao Jia, Chenjie Gu, Qiao Zhang, Jordan Grimstad, Ale Jakse Hartman, Xavier Garcia, Thanumalayan Sankaranarayana Pillai, Jacob Devlin, Michael Laskin, Diego de Las Casas, Dasha Valter, Connie Tao, Lorenzo Blanco, Adrià Puigdomènech Badia, David Reitter, Mianna Chen, Jenny Brennan, Clara Rivera, Sergey Brin, Shariq Iqbal, Gabriela Surita, Jane Labanowski, Abhi Rao, Stephanie Winkler, Emilio Parisotto, Yiming Gu, Kate Olszewska, Ravi Addanki, Antoine Miech, Annie Louis, Denis Teplyashin, Geoff Brown, Elliot Catt, Jan Balaguer, Jackie Xiang, Pidong Wang, Zoe Ashwood, Anton Briukhov, Albert Webson, Sanjay Ganapathy, Smit Sanghavi, Ajay Kannan, Ming-Wei Chang, Axel Stjerngren, Josip Djolonga, Yuting Sun, Ankur Bapna, Matthew Aitchison, Pedram Pejman, Henryk Michalewski, Tianhe Yu, Cindy Wang, Juliette Love, Junwhan Ahn, Dawn Bloxwich, Kehang Han, Peter Humphreys, Thibault Sellam, James Bradbury, Varun Godbole, Sina Samangooei, Bogdan Damoc, Alex Kaskasoli, Sébastien M. R. Arnold, Vijay Vasudevan, Shubham Agrawal, Jason Riesa, Dmitry Lepikhin, Richard Tanburn, Srivatsan Srinivasan, Hyeontaek Lim, Sarah Hodkinson, Pranav Shyam, Johan Ferret, Steven Hand, Ankush Garg, Tom Le Paine, Jian Li, Yujia

Li, Minh Giang, Alexander Neitz, Zaheer Abbas, Sarah York, Machel Reid, Elizabeth Cole, Aakanksha Chowdhery, Dipanjan Das, Dominika Rogozińska, Vitaliy Nikolaev, Pablo Sprechmann, Zachary Nado, Lukas Zilka, Flavien Prost, Luheng He, Marianne Monteiro, Gaurav Mishra, Chris Welty, Josh Newlan, Dawei Jia, Miltiadis Allamanis, Clara Huiyi Hu, Raoul de Liedekerke, Justin Gilmer, Carl Saroufim, Shruti Rijhwani, Shaobo Hou, Disha Shrivastava, Anirudh Baddepudi, Alex Goldin, Adnan Ozturel, Albin Cassirer, Yunhan Xu, Daniel Sohn, Devendra Sachan, Reinald Kim Amplayo, Craig Swanson, Dessie Petrova, Shashi Narayan, Arthur Guez, Siddhartha Brahma, Jessica Landon, Miteyan Patel, Ruizhe Zhao, Kevin Villela, Luyu Wang, Wenhao Jia, Matthew Rahtz, Mai Giménez, Legg Yeung, James Keeling, Petko Georgiev, Diana Mincu, Boxi Wu, Salem Haykal, Rachel Saputro, Kiran Vodrahalli, James Qin, Zeynep Cankara, Abhanshu Sharma, Nick Fernando, Will Hawkins, Behnam Neyshabur, Solomon Kim, Adrian Hutter, Priyanka Agrawal, Alex Castro-Ros, George van den Driessche, Tao Wang, Fan Yang, Shuo yiin Chang, Paul Komarek, Ross McIlroy, Mario Lučić, Guodong Zhang, Wael Farhan, Michael Sharman, Paul Natsev, Paul Michel, Yamini Bansal, Siyuan Qiao, Kris Cao, Siamak Shakeri, Christina Butterfield, Justin Chung, Paul Kishan Rubenstein, Shivani Agrawal, Arthur Mensch, Kedar Soparkar, Karel Lenc, Timothy Chung, Aedan Pope, Loren Maggiore, Jackie Kay, Priya Jhakra, Shibo Wang, Joshua Maynez, Mary Phuong, Taylor Tobin, Andrea Tacchetti, Maja Trebacz, Kevin Robinson, Yash Katariya, Sebastian Riedel, Paige Bailey, Kefan Xiao, Nimesh Ghelani, Lora Aroyo, Ambrose Slone, Neil Houlsby, Xuehan Xiong, Zhen Yang, Elena Gribovskaya, Jonas Adler, Mateo Wirth, Lisa Lee, Music Li, Thais Kagohara, Jay Pavagadhi, Sophie Bridgers, Anna Bortsova, Sanjay Ghemawat, Zafarali Ahmed, Tianqi Liu, Richard Powell, Vijay Bolina, Mariko Iinuma, Polina Zablotskaia, James Besley, Da-Woon Chung, Timothy Dozat, Ramona Comanescu, Xiance Si, Jeremy Greer, Guolong Su, Martin Polacek, Raphaël Lopez Kaufman, Simon Tokumine, Hexiang Hu, Elena Buchatskaya, Yingjie Miao, Mohamed Elhawaty, Aditya Siddhant, Nenad Tomasev, Jinwei Xing, Christina Greer, Helen Miller, Shereen Ashraf, Aurko Roy, Zizhao Zhang, Ada Ma, Angelos Filos, Milos Besta, Rory Blevins, Ted Klimenko, Chih-Kuan Yeh, Soravit Changpinyo, Jiaqi Mu, Oscar Chang, Mantas Pajarskas, Carrie Muir, Vered Cohen, Charline Le Lan, Krishna Haridasan, Amit Marathe, Steven Hansen, Sholto Douglas, Rajkumar Samuel, Mingqiu Wang, Sophia Austin, Chang Lan, Jiepu Jiang, Justin Chiu, Jaime Alonso Lorenzo, Lars Lowe Sjösund, Sébastien Cevey, Zach Gleicher, Thi Avrahami, Anudhyan Boral, Hansa Srinivasan, Vittorio Selo, Rhys May, Konstantinos Aisopos, Léonard Hussenot, Livio Baldini Soares, Kate Baumli, Michael B. Chang, Adrià Recasens, Ben Caine,

Alexander Pritzel, Filip Pavetic, Fabio Pardo, Anita Gergely, Justin Frye, Vinay Ramasesh, Dan Horgan, Kartikeya Badola, Nora Kassner, Subhrajit Roy, Ethan Dyer, Víctor Campos Campos, Alex Tomala, Yunhao Tang, Dalia El Badawy, Elspeth White, Basil Mustafa, Oran Lang, Abhishek Jindal, Sharad Vikram, Zhitao Gong, Sergi Caelles, Ross Hemsley, Gregory Thornton, Fangxiaoyu Feng, Wojciech Stokowiec, Ce Zheng, Phoebe Thacker, Çağlar Ünlü, Zhishuai Zhang, Mohammad Saleh, James Svensson, Max Bileschi, Piyush Patil, Ankesh Anand, Roman Ring, Katerina Tsihlas, Arpi Vezer, Marco Selvi, Toby Shevlane, Mikel Rodriguez, Tom Kwiatkowski, Samira Daruki, Keran Rong, Allan Dafoe, Nicholas FitzGerald, Keren Gu-Lemberg, Mina Khan, Lisa Anne Hendricks, Marie Pellat, Vladimir Feinberg, James Cobon-Kerr, Tara Sainath, Maribeth Rauh, Sayed Hadi Hashemi, Richard Ives, Yana Hasson, Eric Noland, Yuan Cao, Nathan Byrd, Le Hou, Qingze Wang, Thibault Sottiaux, Michela Paganini, Jean-Baptiste Lespiau, Alexandre Moufarek, Samer Hassan, Kaushik Shivakumar, Joost van Amersfoort, Amol Mandhane, Pratik Joshi, Anirudh Goyal, Matthew Tung, Andrew Brock, Hannah Sheahan, Vedant Misra, Cheng Li, Nemanja Rakićević, Mostafa Dehghani, Fangyu Liu, Sid Mittal, Junhyuk Oh, Seb Noury, Eren Sezener, Fantine Huot, Matthew Lamm, Nicola De Cao, Charlie Chen, Sidharth Mudgal, Romina Stella, Kevin Brooks, Gautam Vasudevan, Chenxi Liu, Mainak Chain, Nivedita Melinkeri, Aaron Cohen, Venus Wang, Kristie Seymore, Sergey Zubkov, Rahul Goel, Summer Yue, Sai Krishnakumaran, Brian Albert, Nate Hurley, Motoki Sano, Anhad Mohananey, Jonah Joughin, Egor Filonov, Tomasz Kępa, Yomna Eldawy, Jiawern Lim, Rahul Rishi, Shirin Badiezadegan, Taylor Bos, Jerry Chang, Sanil Jain, Sri Gayatri Sundara Padmanabhan, Subha Puttagunta, Kalpesh Krishna, Leslie Baker, Norbert Kalb, Vamsi Bedapudi, Adam Kurzrok, Shuntong Lei, Anthony Yu, Oren Litvin, Xiang Zhou, Zhichun Wu, Sam Sobell, Andrea Siciliano, Alan Papir, Robby Neale, Jonas Bragagnolo, Tej Toor, Tina Chen, Valentin Anklin, Feiran Wang, Richie Feng, Milad Gholami, Kevin Ling, Lijuan Liu, Jules Walter, Hamid Moghaddam, Arun Kishore, Jakub Adamek, Tyler Mercado, Jonathan Mallinson, Siddhinita Wandekar, Stephen Cagle, Eran Ofek, Guillermo Garrido, Clemens Lombriser, Maksim Mukha, Botu Sun, Hafeezul Rahman Mohammad, Josip Matak, Yadi Qian, Vikas Peswani, Pawel Janus, Quan Yuan, Leif Schelin, Oana David, Ankur Garg, Yifan He, Oleksii Duzhyi, Anton Älgmyr, Timothée Lottaz, Qi Li, Vikas Yadav, Luyao Xu, Alex Chinien, Rakesh Shivanna, Aleksandr Chuklin, Josie Li, Carrie Spadine, Travis Wolfe, Kareem Mohamed, Subhabrata Das, Zihang Dai, Kyle He, Daniel von Dincklage, Shyam Upadhyay, Akanksha Maurya, Luyan Chi, Sebastian Krause, Khalid Salama, Pam G Rabinovitch, Pavan Kumar Reddy M, Aarush Selvan, Mikhail Dektiarev, Golnaz Ghiasi, Erdem Gu-

ven, Himanshu Gupta, Boyi Liu, Deepak Sharma, Idan Heimlich Shtacher, Shachi Paul, Oscar Akerlund, François-Xavier Aubet, Terry Huang, Chen Zhu, Eric Zhu, Elico Teixeira, Matthew Fritze, Francesco Bertolini, Liana-Eleonora Marinescu, Martin Bölle, Dominik Paulus, Khyatti Gupta, Tejasi Latkar, Max Chang, Jason Sanders, Roopa Wilson, Xuewei Wu, Yi-Xuan Tan, Lam Nguyen Thiet, Tulsee Doshi, Sid Lall, Swaroop Mishra, Wanming Chen, Thang Luong, Seth Benjamin, Jasmine Lee, Ewa Andrejczuk, Dominik Rabiej, Vipul Ranjan, Krzysztof Styrc, Pengcheng Yin, Jon Simon, Malcolm Rose Harriott, Mudit Bansal, Alexei Robsky, Geoff Bacon, David Greene, Daniil Mirylenka, Chen Zhou, Obaid Sarvana, Abhimanyu Goyal, Samuel Andermatt, Patrick Siegler, Ben Horn, Assaf Israel, Francesco Pongetti, Chih-Wei "Louis" Chen, Marco Selvatici, Pedro Silva, Kathie Wang, Jackson Tolins, Kelvin Guu, Roey Yogev, Xiaochen Cai, Alessandro Agostini, Maulik Shah, Hung Nguyen, Noah Ó Donnaile, Sébastien Pereira, Linda Friso, Adam Stambler, Adam Kurzrok, Chenkai Kuang, Yan Romanikhin, Mark Geller, ZJ Yan, Kane Jang, Cheng-Chun Lee, Wojciech Fica, Eric Malmi, Qijun Tan, Dan Banica, Daniel Balle, Ryan Pham, Yanping Huang, Diana Avram, Hongzhi Shi, Jasjot Singh, Chris Hidey, Niharika Ahuja, Pranab Saxena, Dan Dooley, Srividya Pranavi Potharaju, Eileen O'Neill, Anand Gokulchandran, Ryan Foley, Kai Zhao, Mike Dusenberry, Yuan Liu, Pulkit Mehta, Ragha Kotikalapudi, Chalence Safranek-Shrader, Andrew Goodman, Joshua Kessinger, Eran Globen, Prateek Kolhar, Chris Gorgolewski, Ali Ibrahim, Yang Song, Ali Eichenbaum, Thomas Brovelli, Sahitya Potluri, Preethi Lahoti, Cip Baetu, Ali Ghorbani, Charles Chen, Andy Crawford, Shalini Pal, Mukund Sridhar, Petru Gurita, Asier Mujika, Igor Petrovski, Pierre-Louis Cedoz, Chenmei Li, Shiyuan Chen, Niccolò Dal Santo, Siddharth Goyal, Jitesh Punjabi, Karthik Kappaganthu, Chester Kwak, Pallavi LV, Sarmishta Velury, Himadri Choudhury, Jamie Hall, Premal Shah, Ricardo Figueira, Matt Thomas, Minjie Lu, Ting Zhou, Chintu Kumar, Thomas Jurdi, Sharat Chikkerur, Yenai Ma, Adams Yu, Soo Kwak, Victor Ähdel, Sujeevan Rajayogam, Travis Choma, Fei Liu, Aditya Barua, Colin Ji, Ji Ho Park, Vincent Hellendoorn, Alex Bailey, Taylan Bilal, Huanjie Zhou, Mehrdad Khatir, Charles Sutton, Wojciech Rzadkowski, Fiona Macintosh, Konstantin Shagin, Paul Medina, Chen Liang, Jinjing Zhou, Pararth Shah, Yingying Bi, Attila Dankovics, Shipra Banga, Sabine Lehmann, Marissa Bredesen, Zifan Lin, John Eric Hoffmann, Jonathan Lai, Raynald Chung, Kai Yang, Nihal Balani, Arthur Bražinskas, Andrei Sozanschi, Matthew Hayes, Héctor Fernández Alcalde, Peter Makarov, Will Chen, Antonio Stella, Liselotte Snijders, Michael Mandl, Ante Kärrman, Paweł Nowak, Xinyi Wu, Alex Dyck, Krishnan Vaidyanathan, Raghavender R, Jessica Mallet, Mitch Rudominer, Eric Johnston, Sushil Mittal, Akhil Udathu, Janara Christensen, Vishal

Verma, Zach Irving, Andreas Santucci, Gamaleldin Elsayed, Elnaz Davoodi, Marin Georgiev, Ian Tenney, Nan Hua, Geoffrey Cideron, Edouard Leurent, Mahmoud Alnahlawi, Ionut Georgescu, Nan Wei, Ivy Zheng, Dylan Scandinaro, Heinrich Jiang, Jasper Snoek, Mukund Sundararajan, Xuezhi Wang, Zack Ontiveros, Itay Karo, Jeremy Cole, Vinu Rajashekhar, Lara Tumeh, Eyal Ben-David, Rishub Jain, Jonathan Uesato, Romina Datta, Oskar Bunyan, Shimu Wu, John Zhang, Piotr Stanczyk, Ye Zhang, David Steiner, Subhajit Naskar, Michael Azzam, Matthew Johnson, Adam Paszke, Chung-Cheng Chiu, Jaume Sanchez Elias, Afroz Mohiuddin, Faizan Muhammad, Jin Miao, Andrew Lee, Nino Vieillard, Jane Park, Jiageng Zhang, Jeff Stanway, Drew Garmon, Abhijit Karmarkar, Zhe Dong, Jong Lee, Aviral Kumar, Luowei Zhou, Jonathan Evens, William Isaac, Geoffrey Irving, Edward Loper, Michael Fink, Isha Arkatkar, Nanxin Chen, Izhak Shafran, Ivan Petrychenko, Zhe Chen, Johnson Jia, Anselm Levskaya, Zhenkai Zhu, Peter Grabowski, Yu Mao, Alberto Magni, Kaisheng Yao, Javier Snaider, Norman Casagrande, Evan Palmer, Paul Suganthan, Alfonso Castaño, Irene Giannoumis, Wooyeol Kim, Mikołaj Rybiński, Ashwin Sreevatsa, Jennifer Prendki, David Soergel, Adrian Goedeckemeyer, Willi Gierke, Mohsen Jafari, Meenu Gaba, Jeremy Wiesner, Diana Gage Wright, Yawen Wei, Harsha Vashisht, Yana Kulizhskaya, Jay Hoover, Maigo Le, Lu Li, Chimezie Iwuanyanwu, Lu Liu, Kevin Ramirez, Andrey Khorlin, Albert Cui, Tian LIN, Marcus Wu, Ricardo Aguilar, Keith Pallo, Abhishek Chakladar, Ginger Perng, Elena Allica Abellan, Mingyang Zhang, Ishita Dasgupta, Nate Kushman, Ivo Penchev, Alena Repina, Xihui Wu, Tom van der Weide, Priya Ponnapalli, Caroline Kaplan, Jiri Simsa, Shuangfeng Li, Olivier Dousse, Fan Yang, Jeff Piper, Nathan Ie, Rama Pasumarthi, Nathan Lintz, Anitha Vijayakumar, Daniel Andor, Pedro Valenzuela, Minnie Lui, Cosmin Paduraru, Daiyi Peng, Katherine Lee, Shuyuan Zhang, Somer Greene, Duc Dung Nguyen, Paula Kurylowicz, Cassidy Hardin, Lucas Dixon, Lili Janzer, Kiam Choo, Ziqiang Feng, Biao Zhang, Achintya Singhal, Dayou Du, Dan McKinnon, Natasha Antropova, Tolga Bolukbasi, Orgad Keller, David Reid, Daniel Finchelstein, Maria Abi Raad, Remi Crocker, Peter Hawkins, Robert Dadashi, Colin Gaffney, Ken Franko, Anna Bulanova, Rémi Leblond, Shirley Chung, Harry Askham, Luis C. Cobo, Kelvin Xu, Felix Fischer, Jun Xu, Christina Sorokin, Chris Alberti, Chu-Cheng Lin, Colin Evans, Alek Dimitriev, Hannah Forbes, Dylan Banarse, Zora Tung, Mark Omernick, Colton Bishop, Rachel Sterneck, Rohan Jain, Jiawei Xia, Ehsan Amid, Francesco Piccinno, Xingyu Wang, Praseem Banzal, Daniel J. Mankowitz, Alex Polozov, Victoria Krakovna, Sasha Brown, MohammadHossein Bateni, Dennis Duan, Vlad Firoiu, Meghana Thotakuri, Tom Natan, Matthieu Geist, Ser tan Girgin, Hui Li, Jiayu Ye, Ofir Roval, Reiko Tojo, Michael Kwong, James Lee-Thorp,

Christopher Yew, Danila Sinopalnikov, Sabela Ramos, John Mellor, Abhishek Sharma, Kathy Wu, David Miller, Nicolas Sonnerat, Denis Vnukov, Rory Greig, Jennifer Beattie, Emily Caveness, Libin Bai, Julian Eisenschlos, Alex Korchemniy, Tomy Tsai, Mimi Jasarevic, Weize Kong, Phuong Dao, Zeyu Zheng, Frederick Liu, Fan Yang, Rui Zhu, Tian Huey Teh, Jason Sanmiya, Evgeny Gladchenko, Nejc Trdin, Daniel Toyama, Evan Rosen, Sasan Tavakkol, Linting Xue, Chen Elkind, Oliver Woodman, John Carpenter, George Papamakarios, Rupert Kemp, Sushant Kafle, Tanya Grunina, Rishika Sinha, Alice Talbert, Diane Wu, Denese Owusu-Afriyie, Cosmo Du, Chloe Thornton, Jordi Pont-Tuset, Pradyumna Narayana, Jing Li, Saaber Fatehi, John Wieting, Omar Ajmeri, Benigno Uria, Yeongil Ko, Laura Knight, Amélie Héliou, Ning Niu, Shane Gu, Chenxi Pang, Yeqing Li, Nir Levine, Ariel Stolovich, Rebeca Santamaria-Fernandez, Sonam Goenka, Wenny Yustalim, Robin Strudel, Ali Elqursh, Charlie Deck, Hyo Lee, Zonglin Li, Kyle Levin, Raphael Hoffmann, Dan Holtmann-Rice, Olivier Bachem, Sho Arora, Christy Koh, Soheil Hassas Yeganeh, Siim Põder, Mukarram Tariq, Yanhua Sun, Lucian Ionita, Mojtaba Seyedhosseini, Pouya Tafti, Zhiyu Liu, Anmol Gulati, Jasmine Liu, Xinyu Ye, Bart Chrzaszcz, Lily Wang, Nikhil Sethi, Tianrun Li, Ben Brown, Shreya Singh, Wei Fan, Aaron Parisi, Joe Stanton, Vinod Koverkathu, Christopher A. Choquette-Choo, Yunjie Li, TJ Lu, Abe Ittycheriah, Prakash Shroff, Mani Varadarajan, Sanaz Bahargam, Rob Willoughby, David Gaddy, Guillaume Desjardins, Marco Cornero, Brona Robenek, Bhavishya Mittal, Ben Albrecht, Ashish Shenoy, Fedor Moiseev, Henrik Jacobsson, Alireza Ghaffarkhah, Morgane Rivière, Alanna Walton, Clément Crepy, Alicia Parrish, Zongwei Zhou, Clement Farabet, Carey Radebaugh, Praveen Srinivasan, Claudia van der Salm, Andreas Fidjeland, Salvatore Scellato, Eri Latorre-Chimoto, Hanna Klimczak-Plucińska, David Bridson, Dario de Cesare, Tom Hudson, Piermaria Mendolicchio, Lexi Walker, Alex Morris, Matthew Mauger, Alexey Guseynov, Alison Reid, Seth Odoom, Lucia Loher, Victor Cotruta, Madhavi Yenugula, Dominik Grewe, Anastasia Petrushkina, Tom Duerig, Antonio Sanchez, Steve Yadlowsky, Amy Shen, Amir Globerson, Lynette Webb, Sahil Dua, Dong Li, Surya Bhupatiraju, Dan Hurt, Haroon Qureshi, Ananth Agarwal, Tomer Shani, Matan Eyal, Anuj Khare, Shreyas Rammohan Belle, Lei Wang, Chetan Tekur, Mihir Sanjay Kale, Jinliang Wei, Ruoxin Sang, Brennan Saeta, Tyler Liechty, Yi Sun, Yao Zhao, Stephan Lee, Pandu Nayak, Doug Fritz, Manish Reddy Vuyyuru, John Aslanides, Nidhi Vyas, Martin Wicke, Xiao Ma, Evgenii Eltyshev, Nina Martin, Hardie Cate, James Manyika, Keyvan Amiri, Yelin Kim, Xi Xiong, Kai Kang, Florian Luisier, Nilesh Tripuraneni, David Madras, Mandy Guo, Austin Waters, Oliver Wang, Joshua Ainslie, Jason Baldridge, Han Zhang, Garima Pruthi, Jakob Bauer, Feng Yang, Riham Mansour, Jason Gel-

man, Yang Xu, George Polovets, Ji Liu, Honglong Cai, Warren Chen, XiangHai Sheng, Emily Xue, Sherjil Ozair, Christof Angermueller, Xiaowei Li, Anoop Sinha, Weiren Wang, Julia Wiesinger, Emmanouil Koukoumidis, Yuan Tian, Anand Iyer, Madhu Gurumurthy, Mark Goldenson, Parashar Shah, MK Blake, Hongkun Yu, Anthony Urbanowicz, Jennimaria Palomaki, Chrisantha Fernando, Ken Durden, Harsh Mehta, Nikola Momchev, Elahe Rahimtoroghi, Maria Georgaki, Amit Raul, Sebastian Ruder, Morgan Redshaw, Jinhyuk Lee, Denny Zhou, Komal Jalan, Dinghua Li, Blake Hechtman, Parker Schuh, Milad Nasr, Kieran Milan, Vladimir Mikulik, Juliana Franco, Tim Green, Nam Nguyen, Joe Kelley, Aroma Mahendru, Andrea Hu, Joshua Howland, Ben Vargas, Jeffrey Hui, Kshitij Bansal, Vikram Rao, Rakesh Ghiya, Emma Wang, Ke Ye, Jean Michel Sarr, Melanie Moranski Preston, Madeleine Elish, Steve Li, Aakash Kaku, Jigar Gupta, Ice Pasupat, Da-Cheng Juan, Milan Someswar, Tejvi M., Xinyun Chen, Aida Amini, Alex Fabrikant, Eric Chu, Xuanyi Dong, Amruta Muthal, Senaka Buthpitiya, Sarthak Jauhari, Nan Hua, Urvashi Khandelwal, Ayal Hitron, Jie Ren, Larissa Rinaldi, Shahar Drath, Avigail Dabush, Nan-Jiang Jiang, Harshal Godhia, Uli Sachs, Anthony Chen, Yicheng Fan, Hagai Taitelbaum, Hila Noga, Zhuyun Dai, James Wang, Chen Liang, Jenny Hamer, Chun-Sung Ferng, Chenel Elkind, Aviel Atias, Paulina Lee, Vít Listík, Mathias Carlen, Jan van de Kerkhof, Marcin Pikus, Krunoslav Zaher, Paul Müller, Sasha Zykova, Richard Stefanec, Vitaly Gatsko, Christoph Hirnschall, Ashwin Sethi, Xingyu Federico Xu, Chetan Ahuja, Beth Tsai, Anca Stefanoiu, Bo Feng, Keshav Dhandhania, Manish Katyal, Akshay Gupta, Atharva Parulekar, Divya Pitta, Jing Zhao, Vivaan Bhatia, Yashodha Bhavnani, Omar Alhadlaq, Xiaolin Li, Peter Danenberg, Dennis Tu, Alex Pine, Vera Filippova, Abhipso Ghosh, Ben Limonchik, Bhargava Urala, Chaitanya Krishna Lanka, Derik Clive, Yi Sun, Edward Li, Hao Wu, Kevin Hongtongsak, Ianna Li, Kalind Thakkar, Kuanysh Omarov, Kushal Majmundar, Michael Alverson, Michael Kucharski, Mohak Patel, Mudit Jain, Maksim Zabelin, Paolo Pelagatti, Rohan Kohli, Saurabh Kumar, Joseph Kim, Swetha Sankar, Vineet Shah, Lakshmi Ramachandruni, Xiangkai Zeng, Ben Bariach, Laura Weidinger, Tu Vu, Alek Andreev, Antoine He, Kevin Hui, Sheleem Kashem, Amar Subramanya, Sissie Hsiao, Demis Hassabis, Koray Kavukcuoglu, Adam Sadovsky, Quoc Le, Trevor Strohman, Yonghui Wu, Slav Petrov, Jeffrey Dean, and Oriol Vinyals. Gemini: A family of highly capable multimodal models, 2024.

[121] Kling Team, Jialu Chen, Yuanzheng Ci, Xiangyu Du, Zipeng Feng, Kun Gai, Sainan Guo, Feng Han, Jingbin He, Kang He, Xiao Hu, Xiaohua Hu, Boyuan Jiang, Fangyuan Kong, Hang Li, Jie Li, Qingyu Li, Shen Li, Xiaohan Li, Yan Li, Jiajun Liang, Borui

Liao, Yiqiao Liao, Weihong Lin, Quande Liu, Xiaokun Liu, Yilun Liu, Yuliang Liu, Shun Lu, Hangyu Mao, Yunyao Mao, Haodong Ouyang, Wenyu Qin, Wanqi Shi, Xiaoyu Shi, Lianghao Su, Haozhi Sun, Peiqin Sun, Pengfei Wan, Chao Wang, Chenyu Wang, Meng Wang, Qiulin Wang, Runqi Wang, Xintao Wang, Xuebo Wang, Zekun Wang, Min Wei, Tiancheng Wen, Guohao Wu, Xiaoshi Wu, Zhenhua Wu, Da Xie, Yingtong Xiong, Yulong Xu, Sile Yang, Zikang Yang, Weicai Ye, Ziyang Yuan, Shenglong Zhang, Shuaiyu Zhang, Yuanxing Zhang, Yufan Zhang, Wenzheng Zhao, Ruiliang Zhou, Yan Zhou, Guosheng Zhu, and Yongjie Zhu. Kling-omni technical report, 2025.

[122] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[123] TW {Van der Schaaf} and Lisette Kanse. Error recovery in socio-technical systems. In J.M. Hoc, P. Millot, E. Hollnagel, and P.C. Cacciabue, editors, *7th European Conference on Cognitive Science Approaches to Process Control (CSAPC '99), Villeneuve d'Asq, France*, pages 151–156. Presses Universitaires de Valenciennes, 1999.

[124] Angeliek van Hout. Acquiring perfectivity and telicity in dutch, italian and polish. *Lingua*, 118 11:1740–1765, 2008. URL https://www.sciencedirect.com/science/article/pii/S0024384107001520.

[125] Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Neural Information Processing Systems*, 2017.

[126] Zeno Vendler. *Linguistics in Philosophy*. Cornell University Press, Ithaca, N.Y.,, 1967.

[127] Hao Wang, Sendong Zhao, Zewen Qiang, Bing Qin, and Ting Liu. Beyond the answers: Reviewing the rationality of multiple choice question answering for the evaluation of large language models. *ArXiv*, abs/2402.01349, 2024.

[128] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. Videomae v2: Scaling video masked autoencoders with dual masking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14549–14560, June 2023.

[129] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model's perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.

[130] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[131] Xin Wang, Taein Kwon, Mahdi Rad, Bowen Pan, Ishani Chakraborty, Sean Andrist, D. Bohus, Ashley Feniello, Bugra Tekin, F. Frujeri, Neel Joshi, and Marc Pollefeys. Holoassist: an egocentric human interaction dataset for interactive ai assistants in the real world. *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 20213–20224, 2023.

[132] Yaoting Wang, Shengqiong Wu, Yuecheng Zhang, Shuicheng Yan, Ziwei Liu, Jiebo Luo, and Hao Fei. Multimodal chain-of-thought reasoning: A comprehensive survey, 2025.

[133] Bo Wu, Shoubin Yu, Zhenfang Chen, Joshua B Tenenbaum, and Chuang Gan. Star: A benchmark for situated reasoning in real-world videos. In *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[134] Junbin Xiao, Xindi Shang, Angela Yao, and Tat-Seng Chua. Next-qa: Next phase of question-answering to explaining temporal actions. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9777–9786, 2021.

[135] Junbin Xiao, Angela Yao, Yicong Li, and Tat-Seng Chua. Can i trust your answer? visually grounded video question answering. *ArXiv*, abs/2309.01327, 2023.

[136] D. Xu, Zhou Zhao, Jun Xiao, Fei Wu, Hanwang Zhang, Xiangnan He, and Yueting Zhuang. Video question answering via gradually refined attention over appearance and motion. *Proceedings of the 25th ACM international conference on Multimedia*, 2017.

[137] Mengge Xue, Zhenyu Hu, Liqun Liu, Kuo Liao, Shuang Li, Honglin Han, Meng Zhao, and Chengguo Yin. Strengthened symbol binding makes large language models reliable multiple-choice selectors. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4331–4344, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.acl-long.237.

[138] Qian Yang, Weixiang Yan, and Aishwarya Agrawal. Decompose and compare consistency: Measuring vlms' answer reliability via task-decomposition consistency comparison. *ArXiv*, abs/2407.07840, 2024.

[139] Tianxiao Yang. *The Role of Working Memory in Following Instructions*. PhD thesis, University of York, York, UK, 2011. URL https://etheses.whiterose.ac.uk/2117/.

[140] Yuan Yao, Tianyu Yu, Ao Zhang, Chongyi Wang, Junbo Cui, Hongji Zhu, Tianchi Cai, Haoyu Li, Weilin Zhao, Zhihui He, Qianyu Chen, Huarong Zhou, Zhensheng Zou, Haoye Zhang, Shengding Hu, Zhi Zheng, Jie Zhou, Jie Cai, Xu Han, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. Minicpm-v: A gpt-4v level mllm on your phone. *arXiv preprint 2408.01800*, 2024.

[141] Shoubin Yu, Jaemin Cho, Prateek Yadav, and Mohit Bansal. Self-chained image-language model for video localization and question answering, 2023.

[142] Tianyu Yu, Haoye Zhang, Yuan Yao, Yunkai Dang, Da Chen, Xiaoman Lu, Ganqu Cui, Taiwen He, Zhiyuan Liu, Tat-Seng Chua, et al. Rlaif-v: Aligning mllms through open-source ai feedback for super gpt-4v trustworthiness. *arXiv preprint arXiv:2405.17220*, 2024.

[143] Tongtian Yue, Jie Cheng, Longteng Guo, Xingyuan Dai, Zijia Zhao, Xingjian He, Gang Xiong, Yisheng Lv, and Jing Liu. Sc-tune: Unleashing self-consistent referential comprehension in large vision language models. *arXiv preprint arXiv:2403.13263*, 2024.

[144] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11975–11986, 2023.

[145] Tianle Zhang, Langtian Ma, Yuchen Yan, Yuchen Zhang, Kai Wang, Yue Yang, Ziyao Guo, Wenqi Shao, Yang You, Yu Qiao, Ping Luo, and Kaipeng Zhang. Rethinking human evaluation protocol for text-to-video models: Enhancing reliability,reproducibility, and practicality, 2024.

[146] Yedi Zhang, Peter E. Latham, and Andrew M. Saxe. Understanding unimodal bias in multimodal deep linear networks. In *International Conference on Machine Learning*, 2023.

[147] Yi-Fan Zhang, Weichen Yu, Qingsong Wen, Xue Wang, Zhang Zhang, Liang Wang, Rong Jin, and Tien-Ping Tan. Debiasing multimodal large language models. *ArXiv*, abs/2403.05262, 2024.

[148] Yuan Zhang, Fei Xiao, Tao Huang, Chun-Kai Fan, Hongyuan Dong, Jiawen Li, Jiacong Wang, Kuan Cheng, Shanghang Zhang, and Haoyuan Guo. Unveiling the tapestry of consistency in large vision-language models. *ArXiv*, abs/2405.14156, 2024.

[149] Yuanhan Zhang, Jinming Wu, Wei Li, Bo Li, Zejun Ma, Ziwei Liu, and Chunyuan Li. Video instruction tuning with synthetic data, 2024.

[150] Bingchen Zhao, Yongshuo Zong, Letian Zhang, and Timothy M. Hospedales. Benchmarking multi-image understanding in vision and language models: Perception, knowledge, reasoning, and multi-hop reasoning. *ArXiv*, abs/2406.12742, 2024.

[151] Yue Zhao, Ishan Misra, Philipp Krähenbühl, and Rohit Girdhar. Learning video representations from large language models, 2022.

[152] Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Large language models are not robust multiple choice selectors, 2024.

[153] Yaoyao Zhong, Wei Ji, Junbin Xiao, Yicong Li, Wei Deng, and Tat-Seng Chua. Video question answering: Datasets, algorithms and challenges. *ArXiv*, abs/2203.01225, 2022.

[154] Yifan Zhong, Fengshuo Bai, Shaofei Cai, Xuchuan Huang, Zhang Chen, Xiaowei Zhang, Yuanfei Wang, Shaoyang Guo, Tianrui Guan, Ka Nam Lui, Zhiquan Qi, Yitao Liang, Yuanpei Chen, and Yaodong Yang. A survey on vision-language-action models: An action tokenization perspective, 2025.

[155] Yiwu Zhong, Licheng Yu, Yang Bai, Shangwen Li, Xueting Yan, and Yin Li. Learning procedure-aware video representation from instructional videos and their narrations, 2023.

[156] Luowei Zhou, Chenliang Xu, and Jason J Corso. Towards automatic learning of procedures from web instructional videos. In *AAAI Conference on Artificial Intelligence*, pages 7590–7598, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17344.

[157] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos, 2019.

[158] Yongshuo Zong, Tingyang Yu, Bingchen Zhao, Ruchika Chavhan, and Timothy Hospedales. Fool your (vision and) language model with embarrassingly simple permutations. *ArXiv*, abs/2310.01651, 2023.

Appendix A

Supplementary material for diagnostics chapters

A.1 Selection bias calibration in multiple-choice VideoQA (Chapter 3)

A.1.1 Decomposition attacks and prompts (exact formats) for each model

Video-LLaMA

Video-LLaMA aligns a BLIP-2-based visual encoder [64] with LLaMA [122]. It incorporates audio signals using Imagebind [43]. Both the video and audio Q-formers are trained on the Webvid-2M dataset [8] to align the modalities. LLaMA is fine-tuned with visual instructional data.

Video-LLaMA is designed as a conversational model and was likely not specifically trained on MCQA, but rather on open-ended VQA datasets to generate captions, descriptions, and explanatory answers. Consequently, throughout our experiments with different prompt options, it tends to engage in dialogue: it repeats the question, responds with the general phrase "Hello, how can I help you today?", describes the video content, or even asks follow-up questions such as "What do you think about the video?" rather than providing a direct answer from a predefined list of options or option IDs.[1] Making it elicit a clear and concise answer option is quite challenging.

[1] It is noteworthy that the model often uses the following template to explain its answer choice: "The ID of the correct answer: a_i. Explanation: …"

To overcome this, we experimented with various prompts, temperature settings, and token limits, and developed the following strategy:

System prompt:

```
Analyse the video, choose the correct answer in multiple choice question answering.
Give the ID of the correct answer option from the given list of options. Be concise.
```

User prompt:

```
<question> Options: <options>. The ID of the correct answer:
```

We set the *number of beams* to 2 and the *temperature* to 0.8. Additionally, we allowed the model up to 30 attempts to produce an answer that includes an answer ID. We used the 7B parameter model due to resource constraints. For answering, the model selects eight uniformly sampled frames.

Video-LLaVA

Video-LLaVA [69] uses the pre-trained CLIP ViT-L/14 vision encoder [102] and the Vicuna language model [21]. Visual features from the encoder are projected into the same-dimensional space as the textual embeddings via a linear projector.

Video-LLaVA effectively delivers concise answers as just a single option based on a short, general prompt. However, in some cases, the model outputs as an answer option only "A" or responds with "yes" or "no" to a closed question.

For this model, we used the following prompt:

```
<video_tag> Question: <question> Options: <options> The ID of the correct answer is:
Respond with only the correct answer ID. ASSISTANT:
```

Notably, on the completely unfamiliar datasets Perception Test and Video-MME adding an empty option increases the number of responses that cannot be parsed as an answer ID (although this increase is statistically insignificant). Combined with the observation that the model tends to disregard options presented later in the prompt, this suggests that the length of the question affects the model's performance, and tokens closer to the end of the prompt have a lower probability.

The model also selects eight frames from the video by uniform sampling. To conserve computational resources, we used Video-LLaVA-7B with half-precision set by Torch's autocast. The temperature is equal to 1.

SeViLA

SeViLA adopts a selfchaining approach with BLIP-2 [64] for keyframe localization and answering question. Its Localizer uniformly samples 32 frames and selects the 4 most relevant key ones, while the Answerer generates responses based on those frames. The Localizer and Answerer both have Flan-T5 [22] for LLM.

SeViLA does not have any unanswered or unparsable responses because it was specifically tailored for the multiple-choice VQA task: internally, it maps any answer option IDs to {A, B, C, D, E}. However, it is limited to observing at most six answer options. In a side experiment, we shifted all answers to options G, H, I, J, K leaving the first six options empty, and obtained identical results to our other experimental setting, *Empty Answers*. This indicates that the model is completely blind to options beyond F and is likely overfitted to the options {A, B, C, D, E}.

Furthermore, the shape of the confusion matrices for NExT-QA and STAR in Figures A.1 and A.2 shows an unusually accurate understanding of these datasets, unlike those for Perception Test and Video-MME. Since NExT-QA and STAR were released before SeViLA, the model might have been fine-tuned on them. On the other hand, this suggests that the positional bias in this model is much less pronounced.

Another vulnerability of SeViLA is that it uses *argmax* calculated on logit probabilities. However, logits without numerical stabilization can produce `NaN` values when activated via softmax. In such cases, according to *argmax*, the model's answer defaults to a_0. Among the settings relevant for debiasing, such behavior was observed in 22 cases on NExT-QA.

For this model, we used the original prompts: **Localizer prompt (when the model selects 32 frames):**

```
Does the information within the frame provide necessary details to accurately answer
the given question?
```

Answerer prompt (when the model selects the answer based on 4 key frames):

```
Considering information in frames, select the correct answer from the options.
```

The confusion matrices for all models in the default (unmodified) setting in Figures A.1–A.3 provide a quick insight into positional bias: darker blue cells indicate a greater accumulation of answers.

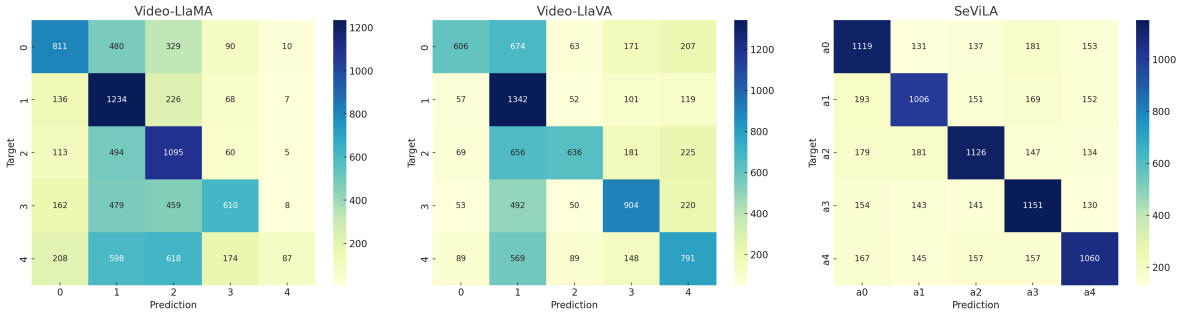


Figure A.1: All Models' Confusion Matrices for NExT-QA.

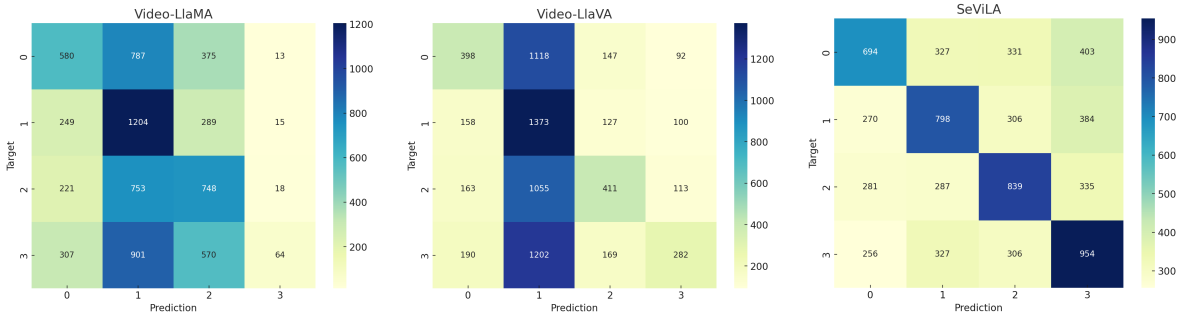


Figure A.2: All Models' Confusion Matrices for STAR.

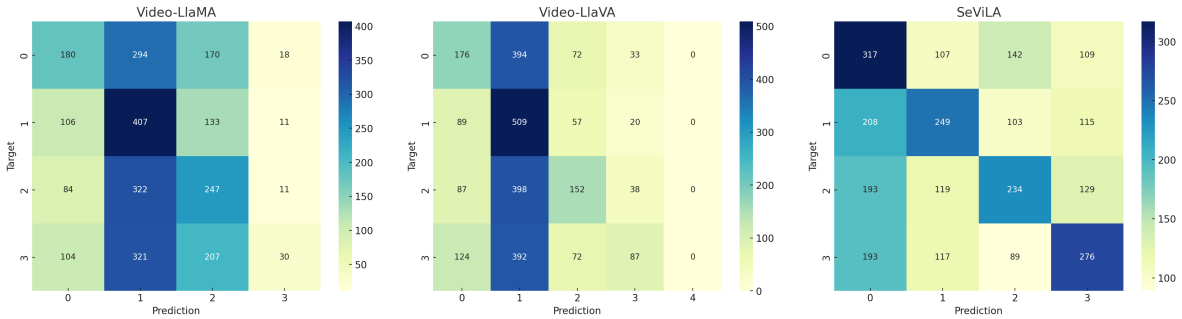


Figure A.3: All Models' Confusion Matrices for Video-MME.

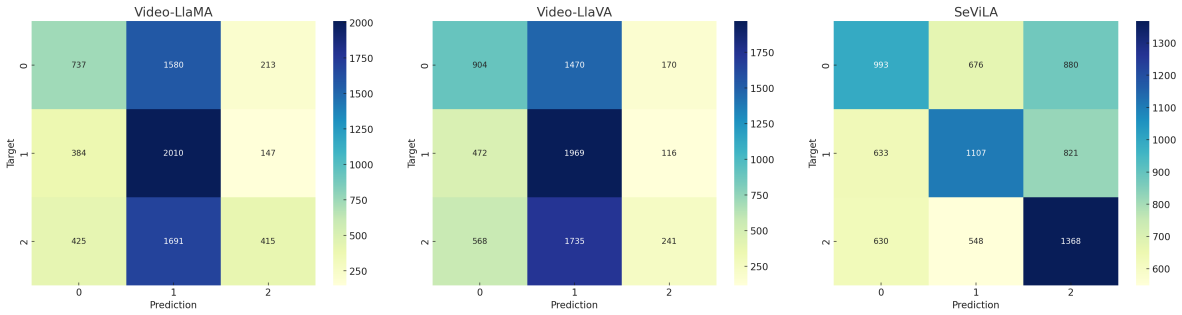


Figure A.4: All Models' Confusion Matrices for Perception Test.

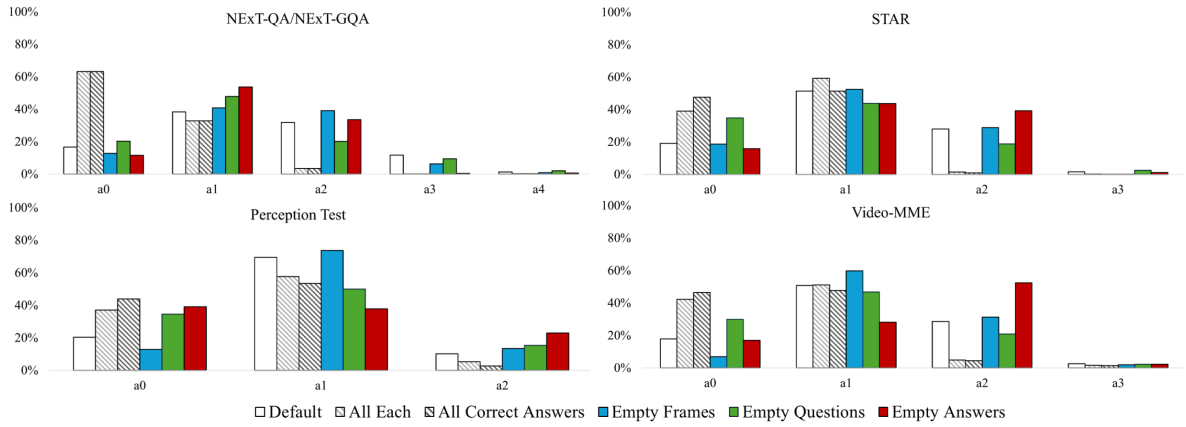


Figure A.5: Video-LLama option distribution for all accuracy-irrelevant settings of NExT-QA, NExT-GQA, STAR, Perception Test and Video-MME.

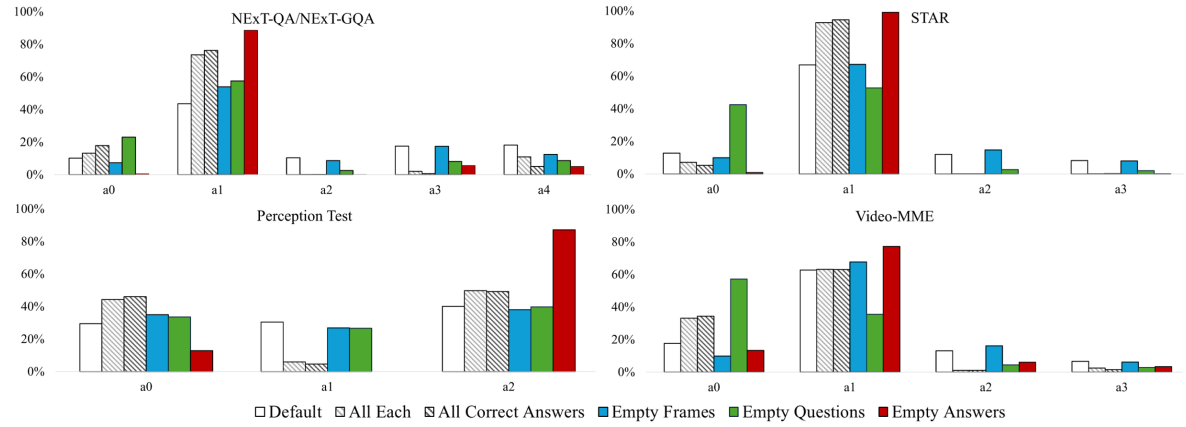


Figure A.6: Video-LLava option distribution for all accuracy-irrelevant settings of NeXT-QA, NExT-GQA, STAR, Perception Test and Video-MME.

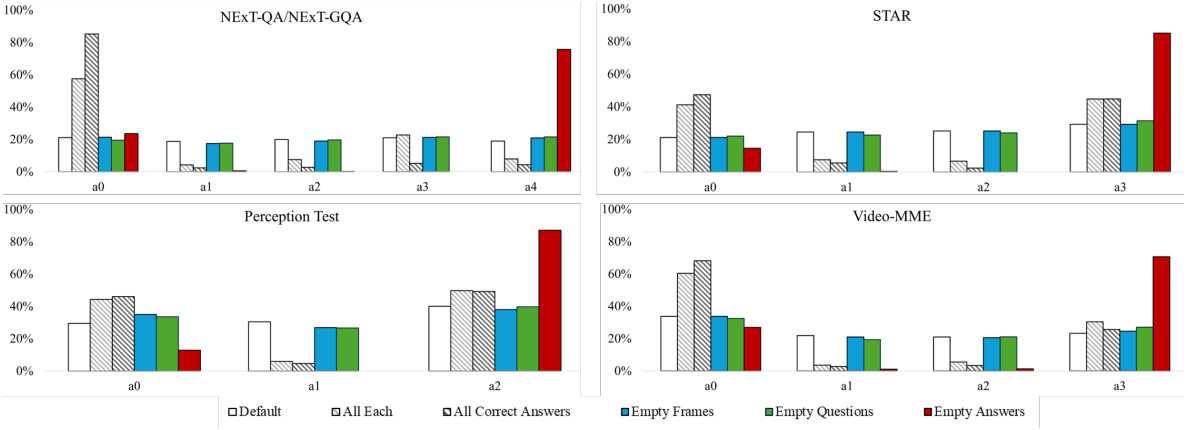


Figure A.7: SeViLA option distribution for all accuracy-irrelevant settings of NExT-QA, NExT-GQA, STAR, Perception Test and Video-MME.

A.1.2 Full benchmark tables (bias metrics + accuracy deltas)

Bias vector visualization

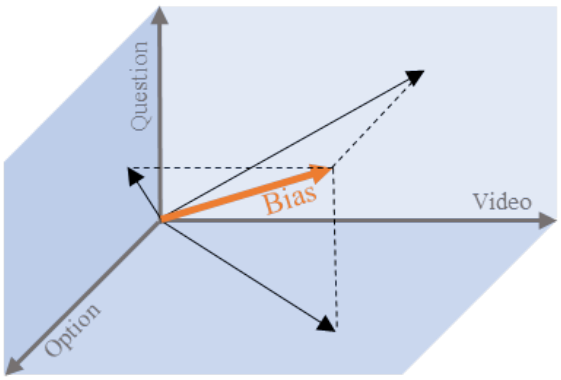


Figure A.8: Bias vector and its projections on the decomposition planes.

A.1.3 Experimental settings examples (question/option modifications)

A.2 CAST: cross-modal self-consistency diagnostics (Chapter 3)

A.2.1 Pair construction and editing operations

As previously mentioned, we sample image pairs from the DOCCI [90] train dataset (10k images), and reject pairs that do not exhibit a certain threshold of CLIP similarity. In particular, we use cosine-similarity between images to filter pairs which are either not similar enough (< 0.75 CLIP score), or which contain near identical images or duplicates (≥ 0.95 CLIP score). We decided on these boundaries through qualitative analysis of the DOCCI samples. Additionally, we filter pairs by description length to only select descriptions with at least 500 characters.

After sampling from our desired image CLIP similarity range, we plot our subset against the text CLIP similarity between each pairs of examples. There is some positive correlation between the similarity of image pairs and the similarity of textual descriptions, as shown in Figure A.9. However, some description similarity can be low even for images pairs which are predicted to be similar. This is because some of the descriptions are short and/or the annotators decided to focus on different aspects of the image.

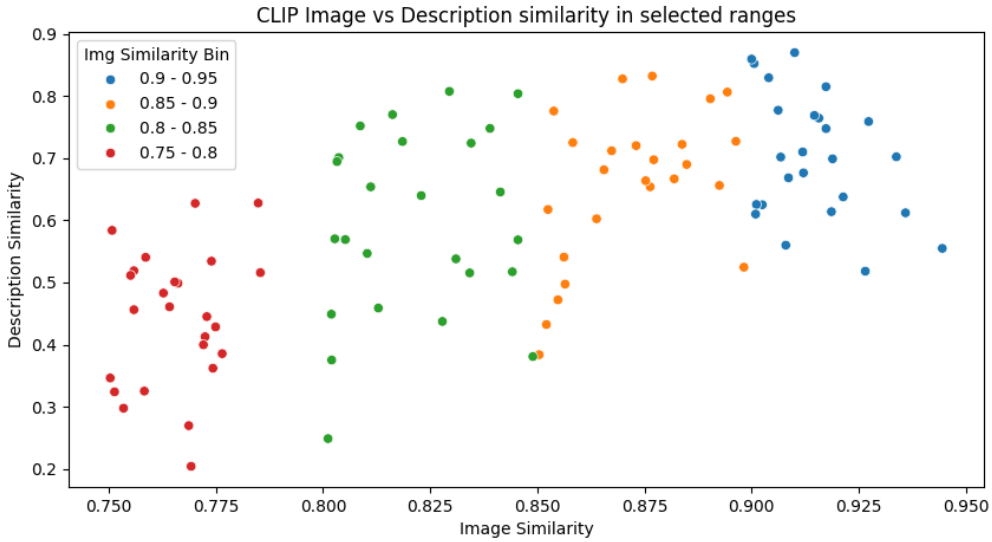


Figure A.9: CLIP similarity between descriptions and images of the sampled example pairs.

A.2.2 Prompts

Generation prompt

To generate a number of similarity statements, we use the prompt shown in Figure A.10. We slightly modify the prompt to fit each modality input.

```
Given two scenes | side-by-side images | scenes and their corresponding images,
find up to five similarities between each scene|image|scene. Output each
similarity in a numbered list.
```

Figure A.10: Generation Prompt: For each model and each of the three modalities, we generate a list of similarity statements using the above prompt.

Evaluation prompt

To reduce variance in our results and potential biases that might exist towards certain prompt phrasing [98, 109], we opt to use three different evaluation prompts shown in Figure A.11.

A.2.3 Results

The full CAST results for all the tested models are shown in Table A.6.

1. Given two scenes|side-by-side images|scenes and their corresponding images, does the following statement apply to only one of the scenes | images | scenes? Answer with 'one' or 'both'.
2. Given two scenes|side-by-side images|scenes and their corresponding images, is the following statement true for both of the scenes|images|scenes? Answer with 'true' or 'false' if the statement is untrue or only true for one of the scenes|images|scenes.
3. Given two descriptions|side-by-side images|descriptions and their corresponding images, does the following statement describe both of the descriptions|images|descriptions? Answer with 'yes' or 'no' if the statement is not applicable to one of the descriptions|images|descriptions.

Figure A.11: Evaluation Prompts: For each model and each of the three modalities, we generate validate a similarity statement from the generation step. We use three different evaluation prompts to reduce potential bias of models towards a particular prompt format.

A.3 Telicity annotations on temporal VideoQA (Chapter 4)

Table A.7 shows results of our telicity test for the temporal split of NExT-QA.

A.4 Scaling temporal VideoQA with broad relation coverage

Paraphrasing prompt

We used the following prompt to create rephrasing of templated questions with LLama2 and GPT-4: `f"Your task is to rephrase this question five times: {question} Do not answer the question, only generate new rephrased options with the same meaning. Here are the rephrased options: 1."`

Examples of generated questions

The examples of Assembly101 templated temporal questions rephrased by Llama2 are shown in Table A.8

A.4.1 Study III (ACL 2025): PERFECT TIMES full materials

This appendix collects the full methodological and experimental material for PERFECT TIMES used in Chapter 4, Section 4.4. It is organized for copy-paste reuse from the paper sources.

A.4.2 Language templates

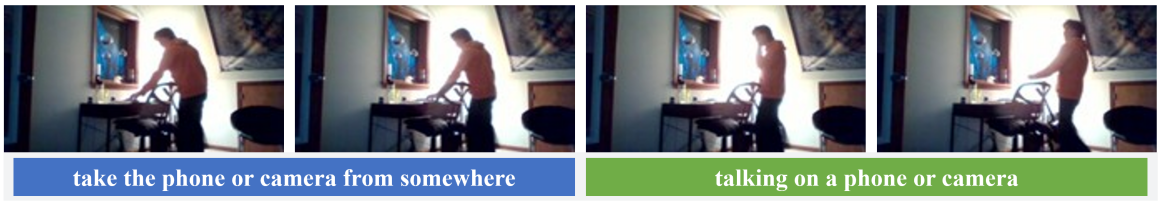
Examples from PERFECT TIMES

The aggregated examples in all languages are presented in Figures A.12-A.17.

Figure A.12: Example from PERFECT TIMES generated by Template 1a for all the languages.

A.5 Models and prompts for testing on PERFECT TIMES

The core idea of VLMs is to bridge the gap between visual features extracted from video and linguistic features from text for such tasks as video captioning, video question answering,



Q: What was the person in the video doing after taking the phone or camera from somewhere?
a0: The person was washing something with a towel.
a1: The person was talking on a phone or camera.
a2: The person talked on a phone or camera.
a3: The person put the phone or camera somewhere.

Q: Cosa faceva la persona nel video dopo aver preso il telefono oppure la fotocamera da qualche parte?
a0: La persona lavava qualcosa con un asciugamano
a1: La persona parlava al telefono oppure alla fotocamera
a2: La persona ha parlato al telefono oppure alla fotocamera
a3: La persona ha messo il telefono oppure la fotocamera da qualche parte

Q: Что делал человек на видео после того, как он взял телефон или камеру откуда-то?
a0: Он протирал что-то полотенцем.
a1: Он разговаривал по телефону или камере.
a2: Он поговорил по телефону или камере.
a3: Он положил телефон или камеру куда-то.

Q: ビデオに写っている人はどこかから携帯電話かカメラを取った後、何をしていましたか？
a0: その人は何かをタオルで洗っていた。
a1: その人は携帯電話かカメラで話していた。
a2: その人は携帯電話かカメラで話した。
a3: その人はどこかに携帯電話かカメラを置いた。

Figure A.13: Example from PERFECT TIMES generated by Template 11a for all the languages.

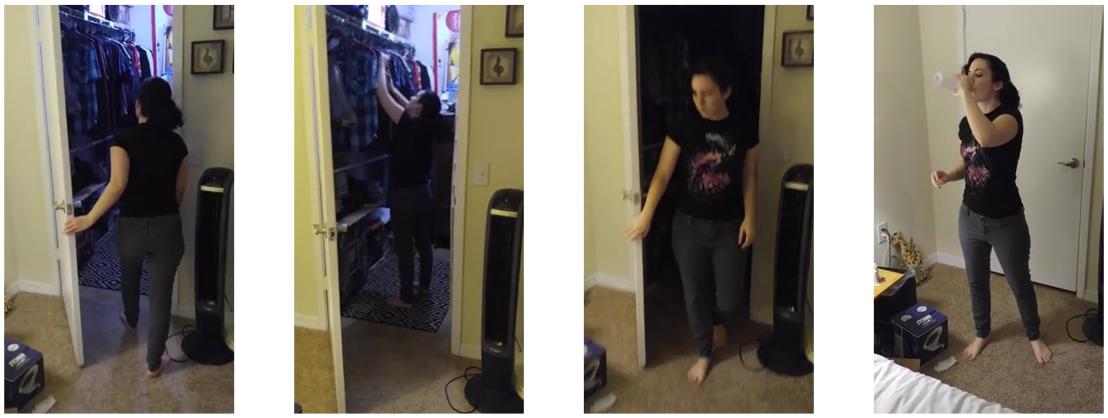
video summarization, and more.

Most VLMs typically share a high-level architecture comprising three main components:

1. **Visual Encoder**: Responsible for extracting meaningful features from the video stream.
2. **Language Encoder (and often a Language Decoder)**: An LLM that handles textual input and generates textual output.
3. **Multimodal Fusion Projector**: This component connects the visual and language parts, allowing them to interact and produce a unified representation. After extracting features from individual frames, it maps these visual features into the same embedding space as the language model's tokens.

A fundamental aspect of VLM processing is that video is generally treated as a sequence of individual images (frames). To prevent the loss of temporal information when processing each frame independently, VLMs often incorporate mechanisms to capture motion and temporal relationships, such as with concatenation of features or through a dedicated temporal transformer.

Frames are sampled either uniformly or by identifying keyframes. For models like GPT-4o, InternVL2, and MiniCPM-V, we use uniform sampling by extracting frames at regular



tiding up the closet | drinking from a cup or glass or bottle

Q: What had the person in the video been doing before drinking from a cup or glass or bottle?

a0: The person had been holding the laptop.

a2: The person had been tidying up the closet or cabinet.

a1: The person had tidied up the closet or cabinet.

a3: The person had closed the closet or cabinet.

Q: Cosa stava facendo la persona nel video prima di bere da una tazza o un bicchiere o una bottiglia?

a0: La persona stava tenendo il laptop

a2: La persona stava riordinando l'armadio oppure i mobili

a1: La persona aveva riordinato l'armadio oppure i mobili

a3: La persona aveva chiuso l'armadio oppure i mobili

Q: Что делал человек на видео до того, как пил из чашки или стакана или бутылки?

a0: Он держал ноутбук.

a2: Он прибирал в шкафу или шкафчике.

a1: Он убрал в шкафу или шкафчике.

a3: Он закрыл шкаф или шкафчик.

Q: ビデオに写っている人はカップかグラスか、またはボトルから飲んでいる前に、何をしていましたか？

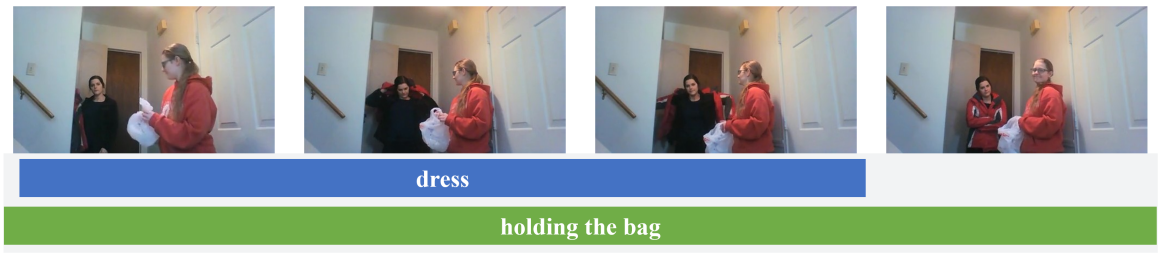
a0: その人はノートパソコンを持っていた

a2: その人はクローゼットかキャビネットを整理していた。

a1: その人はクローゼットかキャビネットを整理した。

a3: その人はクローゼットかキャビネットを閉めた。

Figure A.14: Example from PERFECT TIMES generated by Template 6a for all the languages.



Q: What was the person in the video doing when the other person was dressing?
a0: The person was holding the bag. a2: The person was holding the food.
a1: The person held the bag. a3: The person took the clothes from somewhere.

Q: Cosa faceva la persona nel video quando l'altra persona si stava vestendo?
a0: La persona teneva la borsa a2: La persona teneva il cibo
a1: La persona ha tenuto la borsa a3: La persona ha preso i vestiti da qualche parte

Q: Что делал человек на видео, когда другой человек одевался?
a0: Он держал сумку. a2: Он держал еду.
a1: Он подержал сумку. a3: Он взял одежду откуда-то.

Q: ビデオに写っている人は、他の人が服を着た時、何をしていましたか？
a0: その人はバッグを持っていた。 a2: その人は食べ物を持っていた。
a1: その人はバッグを持った。 a3: その人はどこかから服を取った。

Figure A.15: Example from PERFECT TIMES generated by Template 8b2 for all the languages.

intervals (e.g., every 3 seconds or up to an upper limit). Similarly, Gemini-2.0-Flash-Lite, while inherently multimodal, gets individual frames when it is passed the entire video and interacted with via API. In contrast, LLaVA-NeXT-Video handles video processing internally with its specialized LlavaNextVideoProcessor and Qwen2-VL uses AutoProcessor and Qwen2VLForConditionalGeneration.

The Figure A.18 illustrates the processing pipeline from multimodal input to text generation.

Table A.13 gives details on the open-source models' components.

A.6 Prompts

Since each model accepts input differently, the general question-answer options prompts have been customized.

Qwen2-VL Prompt The Qwen2-VL question messages for all the languages are straightforward, as follows:

"en": question_str = f"Question: {question}" instruction = "Choose the correct answer (a0, a1, a2, or a3)."

Q: What did the person in the video do after they had opened the door?

a0: The person fixed the doorknob.
a1: The person tidied the towel or towels.
a2: The person was opening the door.
a3: The person was fixing the doorknob.

Q: Cosa fece la persona nel video dopo aver aperto la porta?

a0: La persona riparò la maniglia della porta
a1: La persona riordinò l'asciugamano oppure gli asciugamani
a2: La persona stava aprendo la porta
a3: La persona stava riparando la maniglia della porta

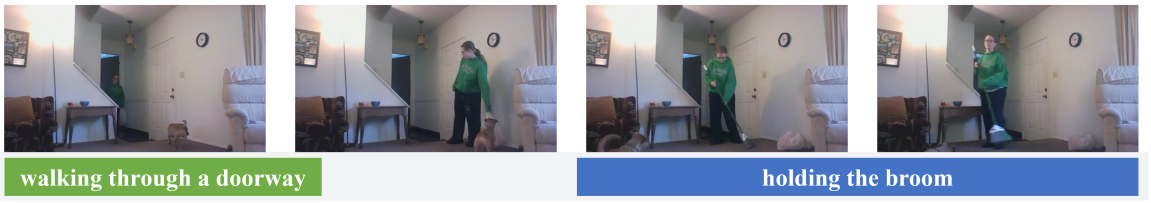
Q: Что сделал человек на видео после того, как открыл дверь?

a0: Он починил дверную ручку.
a1: Он убрал полотенце или полотенца.
a2: Он открывал дверь.
a3: Он чинил дверную ручку.

Q: ビデオに写っている人はくしゃみをしながら、何をしていましたか？

a0: その人は服を持った。
a1: その人は紙かノートで作業した。
a2: その人は紙かノートで作業していた。
a3: その人はどこかにほうきを置いていた。

Figure A.16: Example from PERFECT TIMES generated by Template 2a1 for all the languages.



Q: What had the person in the video done before holding the broom?
a0: The person had taken off the shoes.
a1: The person had been putting the groceries somewhere.
a2: The person had been walking through a doorway.
a3: The person had walked through a doorway.

Q: Cosa aveva fatto la persona nel video prima di tenere la scopa?
a0: La persona si era tolta le scarpe
a1: La persona stava mettendo la spesa da qualche parte
a2: La persona stava passando attraverso un'apertura
a3: La persona era passata attraverso un'apertura

Q: Что сделал человек на видео до того, как он держал швабру?
a0: Он снял обувь.
a1: Он клал продукты куда-то.
a2: Он шел через дверной проем.
a3: Он прошел через дверной проем.

Q: ビデオに写っている人はほうきを持っている前に、何をしましたか？
a0: その人は靴を脱いだ。
a1: その人はどこかに食料品を置いていた。
a2: その人はドアを通っていた。
a3: その人はドアを通った。

Figure A.17: Example from PERFECT TIMES generated by Template 3a for all the languages.

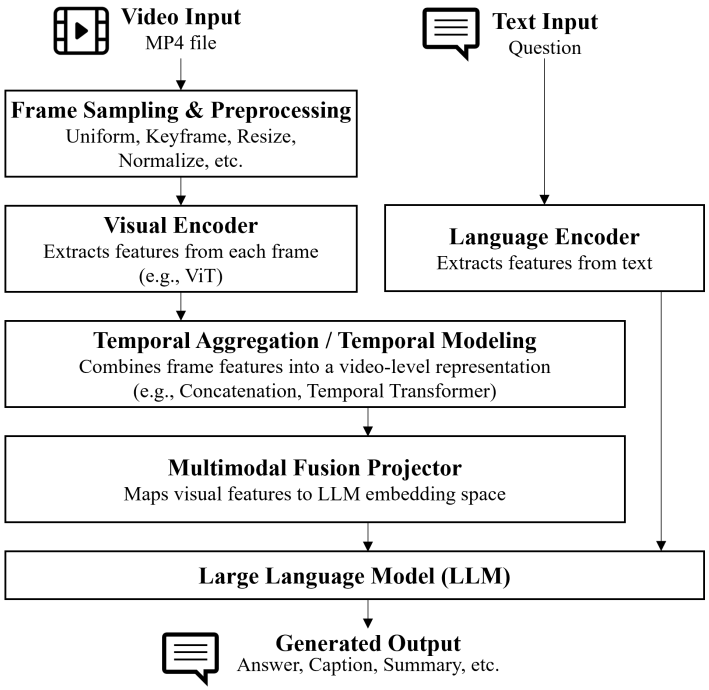


Figure A.18: Overview of a Typical VLM Architecture from raw video and text inputs through encoding, multimodal fusion, and final text generation.

“it”: question_str = f“Domanda: {question}” instruction = “Scegli la risposta corretta (a0, a1, a2 o a3).”

“ru”: question_str = f“Вопрос: {question}” instruction = “Выберите правильный ответ (a0, a1, a2 или a3).”

“jp”: question_str = f“質問: {question}” instruction = “正しい答えを選んでください (a0, a1, a2, a3)。”

Together with the answer options they form text prompts:

```
text\_prompt = (
    f‘‘{question_str}\n’’
    f‘‘a0: {options[’a0’]}\n’’
    f‘‘a1: {options[’a1’]}\n’’
    f‘‘a2: {options[’a2’]}\n’’
    f‘‘a3: {options[’a3’]}\n’’
    f‘‘{instruction}’’
```

The combined input to the models includes videos as follows:

```
messages = [
    {‘‘role’’: ‘‘user’’,
    ‘‘content’’: [
        {‘‘type’’: ‘‘video’’,
        ‘‘video’’: video_uri,
        ‘‘fps’’: 1.0,
        ‘‘max_pixels’’: 360 * 420},
        {‘‘type’’: ‘‘text’’, ‘‘text’’: text_prompt},
        ]
    }
]
```

MiniCPM-V Prompt The text prompt for MiniCPM-V needs specific clarification about the option key. Otherwise it generates an unparsable answer.

“en”: question_str = f“Question: {question}”

instruction = “Choose the correct answer (a0, a1, a2, or a3) and respond only with the option key (e.g., a0). Do not include any additional text.”

“it”: question_str = f“Domanda: {question}”

instruction = “Scegli la risposta corretta (a0, a1, a2 o a3) e rispondi solo con la chiave dell’opzione (es. a0). Non includere testo aggiuntivo.”

“ru”: question_str = f“Вопрос: {question}”

instruction = “Выберите правильный ответ (a0, a1, a2 или a3) и ответь только ключом варианта (например, a0). Не добавляй дополнительный текст.”

“jp”: question_str = f“質問: {question}”

instruction = “正しい答えを選んでください (a0, a1, a2, a3) そしてオプションキーのみで回答してください (例: a0)。追加のテキストを含めないでください。”

The entire text prompt is as follows:

```
text_prompt = (
        f‘‘{question_str}\n’’
        f‘‘a0: {options[’a0’]\n}’’
        f‘‘a1: {options[’a1’]\\n}’’
        f‘‘a2: {options[’a2’]\\n}’’
        f‘‘a3: {options[’a3’]\\n}’’
        f‘‘{instruction}’’)
```

The message content is defined as a list that concatenates the list of video frames (as PIL images) with the text prompt the following way:

```
[{‘‘role’’: ‘‘user’’,
    ‘‘content’’: frames + [text_prompt]}]
```

InternVL2 Prompt The prompt for InternVL2 contains a system message, frame intro, question, answer options and instruction.

```
text_prompt = (
    f‘‘{system_messages[language]}\n’’
    f‘‘{frame_intro[language]}\n\n’’
    f‘‘{question_str}\n’’
    f‘‘a0: {options[’a0’]}\n’’
    f‘‘a1: {options[’a1’]}\n’’
    f‘‘a2: {options[’a2’]}\n’’
    f‘‘a3: {options[’a3’]}\n’’
    f‘‘{instruction}’’
)
```

Localized messages for instructions to InternVL2 are as follows:

system_messages = {

“en”: “Use the provided video frames to answer the question.”,

“it”: “Utilizza i frame del video per rispondere alla domanda.”,

“ru”: “Используй данные кадры видео, чтобы ответить на вопрос.”,

“jp”: “提供されたビデオフレームを使用して質問に答えてください。”

}

frame_intro = {

“en”: “These are the frames from the video.”,
“it”: “Questi sono i frame del video.”,
“ru”: “Это кадры из видео.”,
“jp”: “以下はビデオのフレームです。”
}
“en”: question_str = f“Question: {question}”
instruction = “Choose the correct answer (a0, a1, a2, or a3).”
“it”: question_str = f“Domanda: {question}”
instruction = “Scegli la risposta corretta (a0, a1, a2 o a3).”
“ru”: question_str = f“Вопрос: {question}”
instruction = “Выберите правильный ответ (a0, a1, a2 или a3).”
“jp”: question_str = f“質問: {question}”
instruction = “正しい答えを選んでください (a0, a1, a2, a3)。”

LLaVA-NeXT-Video Prompt LLaVA-NeXT-Video accepts videos as input:

```
{
    ‘‘role’’: ‘‘user’’,
    ‘‘content’’: [
        {‘‘type’’: ‘‘text’’, ‘‘text’’: text_prompt},
        {‘‘type’’: ‘‘video’’,
        ‘‘video’’: f‘‘file://{video_path}’’},
    ],
}
```

The text prompt has the localized question, answer options and instruction as follows:

```
text_prompt = (
        f‘‘{question_str} {question}\n’’
        f‘‘a0: {options[’a0’]}\n’’
        f‘‘a1: {options[’a1’]}\n’’
        f‘‘a2: {options[’a2’]}\n’’
        f‘‘a3: {options[’a3’]}\n’’
        f‘‘{instruction}’’
    )
```

The question messages and instructions in four languages are as follows:
translations = {
“en”: (“Question:”, “Please choose the correct answer (a0, a1, a2, or a3).”),
“it”: (“Domanda:”, “Per favore, scegli la risposta corretta (a0, a1, a2 o a3).”),

"ru": ("Вопрос:", "Пожалуйста, выберите правильный ответ (a0, a1, a2 или a3)."),
"jp": ("質問:", "正しい答えを選んでください (a0, a1, a2, a3)。")
}

GPT-4o Prompt Localized messages for making the GPT-4o prompt are as follows:

system_messages = {
"en": "Use the provided video frames to answer the question.",
"it": "Utilizza i frame del video per rispondere alla domanda.",
"ru": "Используй данные кадры видео, чтобы ответить на вопрос.",
"jp": "提供されたビデオフレームを使用して質問に答えてください。" }
frame_intro = { "en": "These are the frames from the video.",
"it": "Questi sono i frame del video.",
"ru": "Это кадры из видео.",
"jp": "以下はビデオのフレームです。" }

To avoid generating multiple tokens, we further restrict the model in the instruction:

"en": ("Question", "Choose the correct answer (a0, a1, a2, or a3) and respond **only** with the option key (e.g., a0). Do not include any additional text."),

"it": ("Domanda", "Scegli la risposta corretta (a0, a1, a2 o a3) e rispondi **solo** con la chiave dell'opzione (es. a0). Non includere testo aggiuntivo."),

"ru": ("Вопрос", "Выбери правильный ответ (a0, a1, a2 или a3) и ответь **только** ключом варианта (например, a0). Не добавляй дополнительный текст."),

"jp": ("質問", "正しい答えを選んでください (a0, a1, a2, a3) そしてオプションキーのみで回答してください (例: a0)。追加のテキストを含めないでください。"

The entire text prompt contains a question label, a question, answer options and an insruction as follows:

```
text_prompt = (
    f''{question_label}: {question}\n''
    f''a0: {options['a0']}\n''
    f''a1: {options['a1']}\n''
    f''a2: {options['a2']}\n''
    f''a3: {options['a3']}\n''
    f''{instruction}''
)
```

In the input message to the model we pass the text prompt and frames as follows:

```
messages = [
    {''role'': ''system'', ''content'': system_message},
```

```
{‘‘role’’: ‘‘user’’, ‘‘content’’: [frame_message,
*frames_payload, prompt]}
]
```

Gemini-2.0-Flash-Lite Prompt Similarly to GPT-4o, the question previews and instructions for Gemini-2.0-Flash-Lite are localized with the restriction for output tokens as follows:

“en”:

question_str = f“Question: {question}”

instruction = (“Choose the correct answer (a0, a1, a2, or a3) and respond only with the option key (e.g., a0). Do not include any additional text.”)

“it”:

question_str = f“Domanda: {question}”

instruction = (“Scegli la risposta corretta (a0, a1, a2 o a3) e rispondi solo con la chiave dell’opzione (es. a0). Non includere testo aggiuntivo.”)

“ru”:

question_str = f“Вопрос: {question}”

instruction = (“Выбери правильный ответ (a0, a1, a2 или a3) и ответь только ключом варианта (например, a0). Не добавляй дополнительный текст.”)

“jp”:

question_str = f“質問: {question}”

instruction = (“正しい答えを選んでください (a0, a1, a2, a3) そしてオプションキーのみで回答してください (例: a0)。追加のテキストを含めないでください。”)

The text prompt template is the same as for Qwen2-VL and MiniCPM-V:

```
text_prompt = (
        f‘‘{question_str}\n’’
        f‘‘a0: {options[’a0’]}\n’’
        f‘‘a1: {options[’a1’]}\n’’
        f‘‘a2: {options[’a2’]}\n’’
        f‘‘a3: {options[’a3’]}\n’’
        f‘‘{instruction}’’
    )
```

The input contents can simply be the list [text_prompt, video_part], where the video with the proper MIME type, so that the model knows exactly what kind of data it takes.

For all the models the answer is parsed with regex:

```
re.search(r‘‘\ba[0-3]\b’’, generated_text.lower())
```

A.7 Qualitative Error Example

Q: What was the person in the video doing when they put the dish or dishes somewhere?

a0: The person was putting the blanket somewhere.

a1: The person sat down.

a2: The person held the dish.

a3: The person was holding the dish.

Figure A.19: An example from the PERFECT TIMES dataset illustrating the erroneous answer by all the models. The blue and green stripes indicate temporal progression. The correct answer highlighted in bold. The interaction between a completed action (*to put a dish or dishes somewhere*) and a durative action (*to hold the dish*) is justified by the visual modality, as it disambiguates the reference to the different bowls.

Table A.1: Comparison of BOLD and Weighted_BOLD bias mitigation approaches across models and datasets for performance and bias monitoring metrics with $k = 0.25$.

Model	Dataset	Configuration	Performance Metrics		Bias Monitoring Metrics		
			Accuracy	F1_mean	Recall_std	F1_std	JS_std
Video-LLaVA	NExT-QA	**Default**	49.95	49.81	16.31	3.27	5.4
		BOLD	51.81 (↑3.72%)	51.71 (↑3.82%)	13.29 (↓18.54%)	2.66 (↓18.59%)	4.58 (↓15.12%)
		+Weighted_BOLD	51.89 (↑3.88%)	51.77 (↑3.95%)	13.28 (↓18.57%)	2.73 (↓16.57%)	4.56 (↓15.39%)
		-/+Weighted_BOLD	52.28 (↑4.65%)	52.25 (↑4.9%)	10.9 (↓33.18%)	2.23 (↓31.89%)	3.9 (↓27.63%)
	STAR	**Default**	34.77	31.83	24.99	6.82	5.78
		BOLD	37.45 (↑7.7%)	36.14 (↑13.54%)	17.43 (↓30.23%)	4.84 (↓29.02%)	4.52 (↓21.74%)
		+Weighted_BOLD	37.81 (↑8.75%)	36.67 (↑15.22%)	16.15 (↓35.38%)	4.71 (↓30.94%)	4.15 (↓28.15%)
		-/+Weighted_BOLD	39.87 (↑14.67%)	39.65 (↑24.57%)	6.85 (↓72.56%)	2.34 (↓65.67%)	2.08 (↓63.98%)
	Perception Test	**Default**	40.72	35.69	27.5	14.75	5.28
		BOLD	41.62 (↑2.22%)	38.68 (↑8.38%)	21.77 (↓20.85%)	10.93 (↓25.88%)	3.77 (↓28.54%)
		+Weighted_BOLD	41.95 (↑3.02%)	39.46 (↑10.58%)	19.73 (↓28.26%)	10.24 (↓30.53%)	3.39 (↓35.79%)
		-/+Weighted_BOLD	42.01 (↑3.18%)	40.48 (↑13.43%)	14.69 (↓46.59%)	8.48 (↓42.52%)	2.91 (↓44.9%)
	Video-MME	**Default**	34.3	30.99	24.09	8.06	4.79
		BOLD	34.63 (↑0.97%)	32.71 (↑5.54%)	18.21 (↓24.4%)	6.18 (↓23.36%)	3.86 (↓19.47%)
		+Weighted_BOLD	34.59 (↑0.86%)	33.01 (↑6.51%)	15.92 (↓33.93%)	5.91 (↓26.68%)	3.38 (↓29.56%)
		-/+Weighted_BOLD	33.93 (↓1.08%)	33.21 (↑7.16%)	10.1 (↓58.07%)	4.09 (↓49.26%)	2.33 (↓51.39%)
Video-LLaMA	NExT-QA	**Default**	44.79	40.85	23.83	15.86	16.09
		BOLD	45.88 (↑2.43%)	42.15 (↑3.18%)	22.98 (↓3.53%)	15.53 (↓2.11%)	15.56 (↓3.3%)
		+Weighted_BOLD	45.96 (↑2.61%)	42.27 (↑3.46%)	22.78 (↓4.4%)	15.46 (↓2.55%)	15.52 (↓3.55%)
		-/+Weighted_BOLD	47.56 (↑6.19%)	46.63 (↑14.14%)	13.79 (↓42.11%)	8.47 (↓46.62%)	12.14 (↓24.53%)
	STAR	**Default**	36.52	31.86	24.1	15.44	15.29
		BOLD	37.19 (↑1.82%)	33.02 (↑3.65%)	22.29 (↓7.5%)	14.55 (↓5.75%)	14.58 (↓4.6%)
		+Weighted_BOLD	37.21 (↑1.89%)	33.32 (↑4.59%)	21.45 (↓10.97%)	14.03 (↓9.1%)	14.22 (↓7.0%)
		-/+Weighted_BOLD	38.63 (↑5.76%)	38.07 (↑19.52%)	10.16 (↓57.84%)	4.78 (↓69.05%)	8.93 (↓41.61%)
	Perception Test	**Default**	41.39	37.19	27.06	11.4	16.06
		BOLD	41.92 (↑1.27%)	38.7 (↑4.06%)	24.04 (↓11.16%)	9.85 (↓13.61%)	14.44 (↓10.12%)
		+Weighted_BOLD	42.06 (↑1.62%)	39.36 (↑5.86%)	21.8 (↓19.45%)	9.11 (↓20.06%)	13.05 (↓18.75%)
		-/+Weighted_BOLD	42.05 (↑1.59%)	41.35 (↑11.2%)	11.13 (↓58.86%)	4.71 (↓58.65%)	8.43 (↓47.51%)
	Video-MME	**Default**	32.05	28.15	20.62	12.52	13.75
		BOLD	32.69 (↑2.01%)	29.2 (↑3.73%)	19.28 (↓6.5%)	11.95 (↓4.54%)	13.13 (↓4.53%)
		+Weighted_BOLD	32.2 (↑0.47%)	28.98 (↑2.94%)	17.65 (↓14.38%)	11.69 (↓6.56%)	12.6 (↓8.39%)
		-/+Weighted_BOLD	32.92 (↑2.72%)	30.39 (↑7.97%)	15.65 (↓24.08%)	10.21 (↓18.39%)	11.68 (↓15.04%)
SeViLA	NExT-QA	**Default**	63.91	63.88	2.15	1.32	1.99
		BOLD	63.92 (↑0.02%)	63.89 (↑0.03%)	2.0 (↓6.81%)	1.18 (↓10.43%)	1.97 (↓0.91%)
		+Weighted_BOLD	63.88 (↓0.04%)	63.86 (↓0.03%)	2.08 (↓3.47%)	1.28 (↓2.87%)	1.9 (↓4.64%)
		-/+Weighted_BOLD	63.95 (↑0.07%)	63.95 (↑0.11%)	1.44 (↓32.8%)	1.16 (↓12.17%)	1.63 (↓18.3%)
	STAR	**Default**	46.28	46.14	4.47	2.3	2.25
		BOLD	46.17 (↓0.24%)	46.05 (↓0.2%)	4.09 (↓8.42%)	2.24 (↓2.93%)	2.11 (↓6.25%)
		+Weighted_BOLD	46.07 (↓0.46%)	45.95 (↓0.42%)	4.12 (↓7.63%)	2.28 (↓1.24%)	2.13 (↓5.17%)
		-/+Weighted_BOLD	46.2 (↓0.18%)	46.08 (↓0.13%)	4.04 (↓9.49%)	2.23 (↓3.15%)	2.16 (↓3.9%)
	Perception Test	**Default**	45.3	45.11	6.21	3.02	1.59
		BOLD	45.28 (↓0.03%)	45.15 (↑0.09%)	5.1 (↓17.85%)	2.93 (↓3.09%)	1.26 (↓21.04%)
		+Weighted_BOLD	45.34 (↑0.09%)	45.24 (↑0.29%)	4.26 (↓31.39%)	2.65 (↓12.43%)	1.03 (↓35.41%)
		-/+Weighted_BOLD	45.22 (↓0.17%)	45.15 (↑0.09%)	3.69 (↓40.51%)	2.38 (↓21.06%)	0.85 (↓46.67%)
	Video-MME	**Default**	39.85	39.82	4.67	1.68	0.84
		BOLD	40.04 (↑0.46%)	40.04 (↑0.56%)	3.79 (↓18.89%)	1.48 (↓12.03%)	0.74 (↓11.46%)
		+Weighted_BOLD	40.07 (↑0.56%)	40.08 (↑0.66%)	3.64 (↓22.05%)	1.48 (↓11.82%)	0.67 (↓20.11%)
		-/+Weighted_BOLD	40.04 (↑0.46%)	40.06 (↑0.63%)	2.31 (↓50.5%)	1.36 (↓18.83%)	0.3 (↓63.83%)

Table A.2: Comparison of BOLD and Weighted_BOLD bias mitigation approaches across models and datasets for performance and bias monitoring metrics with $k = 0.5$.

Model	Dataset	Configuration	Performance Metrics		Bias Monitoring Metrics		
			Accuracy	F1_mean	Recall_std	F1_std	JS_std
Video-LLaMA	NExT-QA	**Default**	44.79	40.85	23.83	15.86	16.09
		BOLD	45.88 (↑2.43%)	42.15 (↑3.18%)	22.98 (↓3.53%)	15.53 (↓2.11%)	15.55 (↓3.31%)
		+Weighted_BOLD	45.91 (↑2.51%)	42.2 (↑3.29%)	22.83 (↓4.19%)	15.51 (↓2.2%)	15.56 (↓3.27%)
		-	48.25 (↑7.73%)	46.52	16.6	10.6	13.27
		/+Weighted_BOLD		(↑13.86%)	(↓30.32%)	(↓33.17%)	(↓17.48%)
	STAR	**Default**	36.52	31.86	24.1	15.44	15.29
		BOLD	37.19 (↑1.82%)	33.02 (↑3.65%)	22.29 (↓7.5%)	14.55 (↓5.75%)	14.58 (↓4.66%)
		+Weighted_BOLD	37.34 (↑2.24%)	33.5 (↑5.16%)	21.13 (↓12.3%)	14.05 (↓8.98%)	14.14 (↓7.54%)
		-	38.61 (↑5.72%)	38.06	10.18	4.79	8.58
		/+Weighted_BOLD		(↑19.47%)	(↓57.75%)	(↓68.99%)	(↓43.89%)
	Perception Test	**Default**	41.39	37.19	27.06	11.4	16.06
		BOLD	41.9 (↑1.24%)	38.68 (↑4.01%)	24.05 (↓11.11%)	9.87 (↓13.45%)	14.45 (↓10.01%)
		+Weighted_BOLD	42.06 (↑1.62%)	39.36 (↑5.86%)	21.8 (↓19.45%)	9.11 (↓20.06%)	13.02 (↓18.94%)
		-	42.27 (↑2.13%)	41.24	13.55	5.64	9.73
		/+Weighted_BOLD		(↑10.9%)	(↓49.93%)	(↓50.54%)	(↓39.41%)
	Video-MME	**Default**	32.05	28.15	20.62	12.52	13.75
		BOLD	32.73 (↑2.13%)	29.23 (↑3.85%)	19.29 (↓6.45%)	11.96 (↓4.46%)	13.1 (↓4.73%)
		+Weighted_BOLD	32.2 (↑0.47%)	28.98 (↑2.94%)	17.65 (↓14.38%)	11.69 (↓6.56%)	12.64 (↓8.04%)
		-	32.88 (↑2.6%)	31.21	12.77	8.02	10.42
		/+Weighted_BOLD		(↑10.87%)	(↓38.06%)	(↓35.91%)	(↓24.25%)
Video-LLaVA	NExT-QA	**Default**	49.95	49.81	16.31	3.27	5.4
		BOLD	51.81 (↑3.72%)	51.71 (↑3.82%)	13.27 (↓18.63%)	2.67 (↓18.42%)	4.58 (↓15.06%)
		+Weighted_BOLD	52.15 (↑4.39%)	52.04 (↑4.49%)	12.72 (↓22.02%)	2.62 (↓19.83%)	4.49 (↓16.83%)
		-	53.65 (↑7.41%)	53.53	8.49	3.15 (↓3.61%)	3.78
		/+Weighted_BOLD		(↑7.48%)	(↓47.97%)		(↓29.87%)
	STAR	**Default**	34.77	31.83	24.99	6.82	5.78
		BOLD	37.21 (↑7.01%)	35.76 (↑12.37%)	18.28 (↓26.85%)	4.97 (↓27.15%)	4.54 (↓21.39%)
		+Weighted_BOLD	37.53 (↑7.94%)	36.18 (↑13.69%)	17.45 (↓30.15%)	4.91 (↓28.03%)	4.26 (↓26.24%)
		-	38.31 (↑10.17%)	37.39	13.26	4.59	3.25
		/+Weighted_BOLD		(↑17.47%)	(↓46.92%)	(↓32.59%)	(↓43.67%)
	Perception Test	**Default**	40.72	35.69	27.5	14.75	5.28
		BOLD	41.62 (↑2.22%)	38.69 (↑8.4%)	21.75 (↓20.9%)	10.92 (↓25.97%)	3.78 (↓28.4%)
		+Weighted_BOLD	41.95 (↑3.02%)	39.46 (↑10.58%)	19.73 (↓28.26%)	10.24 (↓30.53%)	3.4 (↓35.51%)
		-	41.62 (↑2.22%)	40.88	11.01	5.44	2.59
		/+Weighted_BOLD		(↑14.56%)	(↓59.96%)	(↓63.11%)	(↓50.97%)
	Video-MME	**Default**	34.3	30.99	24.09	8.06	4.79
		BOLD	34.7 (↑1.19%)	32.79 (↑5.81%)	18.19 (↓24.51%)	6.16 (↓23.63%)	3.84 (↓19.93%)
		+Weighted_BOLD	34.63 (↑0.97%)	32.97 (↑6.38%)	16.36 (↓32.07%)	6.01 (↓25.43%)	3.43 (↓28.5%)
		-	34.0 (↓0.86%)	32.6 (↑5.21%)	14.11	5.79	2.92
		/+Weighted_BOLD			(↓41.43%)	(↓28.18%)	(↓39.01%)
SeViLA	NExT-QA	**Default**	63.91	63.88	2.15	1.32	1.99
		BOLD	63.92 (↑0.02%)	63.89 (↑0.02%)	2.03 (↓5.47%)	1.18 (↓10.4%)	1.99 (↓0.17%)
		+Weighted_BOLD	63.93 (↑0.04%)	63.91 (↑0.04%)	1.99 (↓7.64%)	1.19 (↓9.83%)	1.99 (↓0.04%)
		-	64.01 (↑0.16%)	64.0 (↑0.19%)	1.14	1.17	2.01 (↑1.1%)
		/+Weighted_BOLD			(↓47.18%)	(↓11.43%)	
	STAR	**Default**	46.28	46.14	4.47	2.3	2.25
		BOLD	46.22 (↓0.12%)	46.1 (↓0.08%)	4.13 (↓7.46%)	2.26 (↓1.78%)	2.1 (↓6.64%)
		+Weighted_BOLD	46.2 (↓0.18%)	46.08 (↓0.13%)	4.01 (↓10.17%)	2.2 (↓4.36%)	2.07 (↓8.2%)
		-	46.38 (↑0.21%)	46.3 (↑0.34%)	3.41	1.94	1.93
		/+Weighted_BOLD			(↓23.66%)	(↓15.89%)	(↓14.33%)
	Perception Test	**Default**	45.3	45.11	6.21	3.02	1.59
		BOLD	45.32 (↑0.06%)	45.18 (↑0.16%)	5.18 (↓16.61%)	2.95 (↓2.43%)	1.3 (↓18.42%)
		+Weighted_BOLD	45.31 (↑0.03%)	45.2 (↑0.21%)	4.56 (↓26.58%)	2.83 (↓6.43%)	1.08 (↓31.92%)
		-	45.25 (↓0.12%)	45.2 (↑0.2%)	3.25	2.28	0.71
		/+Weighted_BOLD			(↓47.69%)	(↓24.52%)	(↓55.13%)
	Video-MME	**Default**	39.85	39.82	4.67	1.68	0.84
		BOLD	40.19 (↑0.84%)	40.17 (↑0.88%)	4.11 (↓12.03%)	1.41 (↓16.19%)	0.73 (↓12.52%)
		+Weighted_BOLD	40.04 (↑0.46%)	40.03 (↑0.54%)	3.78 (↓19.14%)	1.44 (↓14.6%)	0.69 (↓17.22%)
		-	39.81 (↓0.09%)	39.83	2.99	1.43	0.45
		/+Weighted_BOLD		(↑0.04%)	(↓35.93%)	(↓14.97%)	(↓45.74%)

Table A.3: Comparison of BOLD and Weighted_BOLD bias mitigation approaches across models and datasets for performance and bias monitoring metrics with $k = 0.75$.

Model	Dataset	Configuration	Performance Metrics		Bias Monitoring Metrics		
			Accuracy	F1_mean	Recall_std	F1_std	JS_std
Video-LLaMA	NExT-QA	**Default**	44.79	40.85	23.83	15.86	16.09
		BOLD	45.87 (↑2.4%)	42.16 (↑3.19%)	22.96 (↓3.63%)	15.49 (↓2.37%)	15.56 (↓3.28%)
		+Weighted_BOLD	45.91 (↑2.51%)	42.2 (↑3.29%)	22.83 (↓4.19%)	15.51 (↓2.2%)	15.57 (↓3.23%)
		-	47.56 (↑6.19%)	46.63	13.36	8.42	11.79
		/+Weighted_BOLD		(↑14.14%)	(↓43.91%)	(↓46.91%)	(↓26.73%)
	STAR	**Default**	36.52	31.86	24.1	15.44	15.29
		BOLD	37.19 (↑1.82%)	33.02 (↑3.65%)	22.29 (↓7.5%)	14.55 (↓5.75%)	14.59 (↓4.59%)
		+Weighted_BOLD	37.34 (↑2.24%)	33.5 (↑5.16%)	21.13 (↓12.3%)	14.05 (↓8.98%)	14.13 (↓7.56%)
		-	38.74 (↑6.07%)	38.14	10.58	4.94	9.64
		/+Weighted_BOLD		(↑19.74%)	(↓56.11%)	(↓68.03%)	(↓36.94%)
	Perception Test	**Default**	41.39	37.19	27.06	11.4	16.06
		BOLD	41.9 (↑1.24%)	38.67 (↑3.98%)	24.09 (↓11.0%)	9.88 (↓13.29%)	14.46 (↓9.97%)
		+Weighted_BOLD	42.06 (↑1.62%)	39.36 (↑5.86%)	21.8 (↓19.45%)	9.11 (↓20.06%)	13.12 (↓18.33%)
		-	42.29 (↑2.16%)	41.32	12.96	5.53	9.56
		/+Weighted_BOLD		(↑11.13%)	(↓52.13%)	(↓51.52%)	(↓40.45%)
	Video-MME	**Default**	32.05	28.15	20.62	12.52	13.75
		BOLD	32.69 (↑2.01%)	29.2 (↑3.75%)	19.24 (↓6.7%)	11.94 (↓4.63%)	13.09 (↓4.8%)
		+Weighted_BOLD	32.2 (↑0.47%)	28.97 (↑2.94%)	17.65 (↓14.38%)	11.7 (↓6.54%)	12.56 (↓8.62%)
		-	32.65 (↑1.89%)	30.55	13.94	9.24	11.04
		/+Weighted_BOLD		(↑8.55%)	(↓32.38%)	(↓26.17%)	(↓19.73%)
Video-LLaVA	NExT-QA	**Default**	49.95	49.81	16.31	3.27	5.4
		BOLD	51.81 (↑3.72%)	51.71 (↑3.82%)	13.27 (↓18.63%)	2.67 (↓18.42%)	4.58 (↓15.14%)
		+Weighted_BOLD	51.96 (↑4.02%)	51.86 (↑4.12%)	12.92 (↓20.76%)	2.64 (↓19.44%)	4.48 (↓17.02%)
		-	53.75 (↑7.6%)	53.66	8.51	2.58	3.6 (↓33.28%)
		/+Weighted_BOLD		(↑7.73%)	(↓47.82%)	(↓21.07%)	
	STAR	**Default**	34.77	31.83	24.99	6.82	5.78
		BOLD	37.21 (↑7.01%)	35.76 (↑12.37%)	18.28 (↓26.85%)	4.97 (↓27.15%)	4.62 (↓20.03%)
		+Weighted_BOLD	37.52 (↑7.9%)	36.17 (↑13.64%)	17.45 (↓30.14%)	4.91 (↓28.02%)	4.27 (↓26.1%)
		-	37.86 (↑8.87%)	36.87	13.12	4.68	3.09
		/+Weighted_BOLD		(↑15.85%)	(↓47.51%)	(↓31.37%)	(↓46.45%)
	Perception Test	**Default**	40.72	35.69	27.5	14.75	5.28
		BOLD	41.62 (↑2.22%)	38.68 (↑8.38%)	21.77 (↓20.85%)	10.93 (↓25.88%)	3.79 (↓28.21%)
		+Weighted_BOLD	41.95 (↑3.02%)	39.46 (↑10.58%)	19.73 (↓28.26%)	10.24 (↓30.53%)	3.41 (↓35.4%)
		-	42.21 (↑3.66%)	41.64	9.47	5.14	1.8 (↓65.84%)
		/+Weighted_BOLD		(↑16.67%)	(↓65.56%)	(↓65.14%)	
	Video-MME	**Default**	34.3	30.99	24.09	8.06	4.79
		BOLD	34.63 (↑0.97%)	32.64 (↑5.33%)	18.59 (↓22.84%)	6.19 (↓23.2%)	3.85 (↓19.65%)
		+Weighted_BOLD	34.48 (↑0.54%)	32.89 (↑6.13%)	15.94 (↓33.83%)	5.86 (↓27.32%)	3.37 (↓29.66%)
		-	33.78 (↓1.51%)	33.2 (↑7.12%)	9.23 (↓61.7%)	3.35 (↓58.46%)	1.42 (↓70.35%)
		/+Weighted_BOLD					
SeViLA	NExT-QA	**Default**	63.91	63.88	2.15	1.32	1.99
		BOLD	63.91 (↑0.0%)	63.88 (↑0.01%)	2.03 (↓5.45%)	1.18 (↓10.77%)	1.98 (↓0.34%)
		+Weighted_BOLD	63.93 (↑0.04%)	63.91 (↑0.04%)	2.0 (↓6.83%)	1.2 (↓8.97%)	1.98 (↓0.23%)
		-	64.11 (↑0.31%)	64.09	1.2 (↓44.36%)	1.09	2.15 (↑8.2%)
		/+Weighted_BOLD		(↑0.34%)		(↓17.41%)	
	STAR	**Default**	46.28	46.14	4.47	2.3	2.25
		BOLD	46.27 (↓0.03%)	46.15 (↑0.02%)	4.07 (↓8.91%)	2.22 (↓3.53%)	2.08 (↓7.72%)
		+Weighted_BOLD	46.17 (↓0.24%)	46.05 (↓0.19%)	4.01 (↓10.17%)	2.2 (↓4.43%)	2.04 (↓9.14%)
		-	46.17 (↓0.24%)	46.1 (↓0.09%)	2.96	1.78	1.64
		/+Weighted_BOLD			(↓33.63%)	(↓22.67%)	(↓27.25%)
	Perception Test	**Default**	45.3	45.11	6.21	3.02	1.59
		BOLD	45.31 (↑0.03%)	45.16 (↑0.12%)	5.25 (↓15.42%)	2.98 (↓1.26%)	1.3 (↓18.44%)
		+Weighted_BOLD	45.35 (↑0.12%)	45.25 (↑0.32%)	4.27 (↓31.19%)	2.65 (↓12.18%)	1.03 (↓34.92%)
		-	45.15 (↓0.32%)	45.09	3.48	2.3 (↓24.02%)	0.78
		/+Weighted_BOLD		(↓0.03%)	(↓43.88%)		(↓51.02%)
	Video-MME	**Default**	39.85	39.82	4.67	1.68	0.84
		BOLD	40.04 (↑0.46%)	40.03 (↑0.53%)	3.84 (↓17.85%)	1.46 (↓13.42%)	0.69 (↓18.0%)
		+Weighted_BOLD	39.89 (↑0.09%)	39.87 (↑0.15%)	3.7 (↓20.9%)	1.43 (↓14.96%)	0.58 (↓31.13%)
		-	39.7 (↓0.37%)	39.71	2.69	1.42	0.15
		/+Weighted_BOLD		(↓0.26%)	(↓42.53%)	(↓15.75%)	(↓81.72%)

Table A.4: Comparison of BOLD and Weighted_BOLD bias mitigation approaches across models and datasets for performance and bias monitoring metrics k = 1 (equal to the entire dataset).

Model	Dataset	Configuration	Performance Metrics		Bias Monitoring Metrics		
			Accuracy	F1_mean	Recall_std	F1_std	JS_std
Video-LLaMA	NExT-QA	**Default**	44.79	40.85	23.83	15.86	16.09
		BOLD	45.87 (↑2.4%)	42.16 (↑3.19%)	22.96 (↓3.63%)	15.49 (↓2.37%)	15.56 (↓3.29%)
		+Weighted_BOLD	45.91 (↑2.51%)	42.2 (↑3.29%)	22.83 (↓4.19%)	15.51 (↓2.2%)	15.57 (↓3.23%)
		-/+Weighted_BOLD	47.53 (↑6.11%)	46.62 (↑14.1%)	13.25 (↓44.39%)	8.46 (↓46.68%)	11.78 (↓26.76%)
	STAR	**Default**	36.52	31.86	24.1	15.44	15.29
		BOLD	37.19 (↑1.82%)	33.02 (↑3.65%)	22.29 (↓7.5%)	14.55 (↓5.75%)	14.59 (↓4.58%)
		+Weighted_BOLD	37.34 (↑2.24%)	33.5 (↑5.16%)	21.13 (↓12.3%)	14.05 (↓8.98%)	14.13 (↓7.55%)
		-/+Weighted_BOLD	38.72 (↑6.03%)	38.18 (↑19.85%)	10.0 (↓58.49%)	4.85 (↓68.6%)	8.53 (↓44.23%)
	Perception Test	**Default**	41.39	37.19	27.06	11.4	16.06
		BOLD	41.89 (↑1.21%)	38.67 (↑3.98%)	24.04 (↓11.18%)	9.86 (↓13.52%)	14.45 (↓10.04%)
		+Weighted_BOLD	42.08 (↑1.65%)	39.38 (↑5.89%)	21.8 (↓19.46%)	9.11 (↓20.03%)	12.93 (↓19.5%)
		-/+Weighted_BOLD	42.25 (↑2.07%)	41.53 (↑11.69%)	11.21 (↓58.59%)	4.76 (↓58.2%)	8.83 (↓45.01%)
	Video-MME	**Default**	32.05	28.15	20.62	12.52	13.75
		BOLD	32.65 (↑1.89%)	29.13 (↑3.49%)	19.29 (↓6.44%)	12.03 (↓3.86%)	13.09 (↓4.82%)
		+Weighted_BOLD	32.2 (↑0.47%)	28.97 (↑2.94%)	17.65 (↓14.38%)	11.7 (↓6.54%)	12.57 (↓8.56%)
		-/+Weighted_BOLD	32.92 (↑2.72%)	31.32 (↑11.27%)	12.6 (↓38.89%)	7.85 (↓37.25%)	10.36 (↓24.67%)
Video-LLaVA	NExT-QA	**Default**	49.95	49.81	16.31	3.27	5.4
		BOLD	51.81 (↑3.72%)	51.71 (↑3.82%)	13.27 (↓18.63%)	2.67 (↓18.42%)	4.58 (↓15.18%)
		+Weighted_BOLD	51.83 (↑3.76%)	51.77 (↑3.93%)	12.79 (↓21.58%)	2.5 (↓23.66%)	4.43 (↓17.81%)
		-/+Weighted_BOLD	52.94 (↑5.98%)	53.0 (↑6.41%)	8.49 (↓47.93%)	1.98 (↓39.48%)	3.02 (↓43.96%)
	STAR	**Default**	34.77	31.83	24.99	6.82	5.78
		BOLD	37.14 (↑6.81%)	35.65 (↑12.02%)	18.53 (↓25.84%)	5.01 (↓26.55%)	4.6 (↓20.38%)
		+Weighted_BOLD	37.33 (↑7.37%)	35.98 (↑13.04%)	17.62 (↓29.49%)	4.83 (↓29.09%)	4.35 (↓24.73%)
		-/+Weighted_BOLD	38.56 (↑10.9%)	37.86 (↑18.96%)	11.35 (↓54.56%)	3.69 (↓45.92%)	2.91 (↓49.67%)
	Perception Test	**Default**	40.72	35.69	27.5	14.75	5.28
		BOLD	41.62 (↑2.22%)	38.69 (↑8.4%)	21.75 (↓20.9%)	10.92 (↓25.97%)	3.79 (↓28.14%)
		+Weighted_BOLD	41.95 (↑3.02%)	39.46 (↑10.58%)	19.73 (↓28.26%)	10.24 (↓30.53%)	3.41 (↓35.37%)
		-/+Weighted_BOLD	41.57 (↑2.09%)	41.11 (↑15.2%)	8.95 (↓67.47%)	4.3 (↓70.81%)	2.11 (↓59.94%)
	Video-MME	**Default**	34.3	30.99	24.09	8.06	4.79
		BOLD	34.56 (↑0.76%)	32.63 (↑5.3%)	18.24 (↓24.28%)	6.08 (↓24.59%)	3.86 (↓19.41%)
		+Weighted_BOLD	34.41 (↑0.32%)	32.78 (↑5.77%)	16.1 (↓33.17%)	5.89 (↓26.93%)	3.37 (↓29.69%)
		-/+Weighted_BOLD	34.22 (↓0.22%)	33.56 (↑8.31%)	9.8 (↓59.31%)	3.71 (↓54.0%)	1.84 (↓61.61%)
SeViLA	NExT-QA	**Default**	63.91	63.88	2.15	1.32	1.99
		BOLD	63.91 (↑0.0%)	63.88 (↑0.01%)	2.03 (↓5.45%)	1.18 (↓10.77%)	1.98 (↓0.66%)
		+Weighted_BOLD	64.0 (↑0.15%)	63.98 (↑0.15%)	1.95 (↓9.2%)	1.24 (↓6.37%)	1.97 (↓0.96%)
		-/+Weighted_BOLD	64.07 (↑0.26%)	64.06 (↑0.29%)	0.84 (↓60.74%)	1.07 (↓18.78%)	2.08 (↑4.64%)
	STAR	**Default**	46.28	46.14	4.47	2.3	2.25
		BOLD	46.25 (↓0.06%)	46.14 (↓0.01%)	4.05 (↓9.25%)	2.22 (↓3.75%)	2.06 (↓8.29%)
		+Weighted_BOLD	46.15 (↓0.27%)	46.04 (↓0.22%)	4.01 (↓10.23%)	2.2 (↓4.52%)	2.05 (↓9.07%)
		-/+Weighted_BOLD	46.18 (↓0.21%)	46.1 (↓0.1%)	3.48 (↓22.16%)	2.04 (↓11.64%)	1.85 (↓17.64%)
	Perception Test	**Default**	45.3	45.11	6.21	3.02	1.59
		BOLD	45.4 (↑0.23%)	45.25 (↑0.33%)	5.24 (↓15.53%)	3.04 (↑0.72%)	1.3 (↓18.45%)
		+Weighted_BOLD	45.34 (↑0.09%)	45.24 (↑0.3%)	4.16 (↓33.06%)	2.62 (↓13.42%)	1.03 (↓34.99%)
		-/+Weighted_BOLD	45.22 (↓0.17%)	45.14 (↑0.08%)	3.85 (↓38.0%)	2.43 (↓19.68%)	0.9 (↓43.13%)
	Video-MME	**Default**	39.85	39.82	4.67	1.68	0.84
		BOLD	40.0 (↑0.37%)	39.99 (↑0.44%)	3.81 (↓18.5%)	1.48 (↓11.8%)	0.66 (↓21.49%)
		+Weighted_BOLD	39.81 (↓0.09%)	39.81 (↓0.02%)	3.57 (↓23.49%)	1.45 (↓13.53%)	0.53 (↓36.44%)
		-/+Weighted_BOLD	39.85 (↑0.0%)	39.85 (↑0.08%)	2.66 (↓43.14%)	1.47 (↓12.42%)	0.16 (↓80.65%)

Table A.5: Example of question and answer modifications from the Video-MME benchmark. The correct answer is highlighted in bold in the Default setting.

Modification Type	**Question**	**Answers**
Default	How many national flags appear in the video?	(a0) 3. (a1) 5. (a2) 2. **(a3) 4.**
Empty answers	Default question	(a0) – (a1) – (a2) – (a3) –
Answer shuffling	Default question	(a0) 4. (a1) 5. (a2) 2. (a3) 3.
Additional empty option	Default question	(a0) 3. (a1) 5. (a2) 2. (a3) 4. (a4) –
Correct answer in each option	Default question	(a0) 4. (a1) 4. (a2) 4. (a3) 4.
All identical answers (a0)	Default question	(a0) 3. (a1) 3. (a2) 3. (a3) 3.
All identical answers (a1)	Default question	(a0) 5. (a1) 5. (a2) 5. (a3) 5.
All identical answers (a2)	Default question	(a0) 2. (a1) 2. (a2) 2. (a3) 2.
All identical answers (a3)	Default question	(a0) 4. (a1) 4. (a2) 4. (a3) 4.
Correct answer in (a0)	Default question	(a0) 4. (a1) 5. (a2) 2. (a3) 3.
Correct answer in (a1)	Default question	(a0) 3. (a1) 4. (a2) 2. (a3) 5.
Correct answer in (a2)	Default question	(a0) 3. (a1) 5. (a2) 4. (a3) 2.
Correct answer in (a3)	Default question	(a0) 3. (a1) 5. (a2) 2. (a3) 4.
Correct answer (a0) with shuffling	Default question	(a0) 4. (a1) 5. (a2) 3. (a3) 2.
Correct answer (a1) with shuffling	Default question	(a0) 5. (a1) 4. (a2) 2. (a3) 3.
Correct answer (a2) with shuffling	Default question	(a0) 3. (a1) 2. (a2) 4. (a3) 5.
Correct answer (a3) with shuffling	Default question	(a0) 5. (a1) 3. (a2) 2. (a3) 4.
Rephrased question	What is the total number of national flags that appear in the video?	(a0) 3. (a1) 5. (a2) 2. (a3) 4.
Empty questions	–	(a0) 3. (a1) 5. (a2) 2. (a3) 4.

Model	Gen w/	Eval w/ text				Eval w/ image				Eval w/ both			
		yes/no	both/one	true/false	Avg.	yes/no	both/one	true/false	Avg.	yes/no	both/one	true/false	Avg.
Bunny	text	0.97	0.99	0.84	0.93	0.73	0.88	0.66	0.76	0.99	0.96	0.93	0.96
	image	0.72	0.86	0.55	0.71	0.75	0.90	0.74	0.80	0.85	0.89	0.80	0.85
	both	0.94	0.98	0.80	0.91	0.77	0.92	0.74	0.81	0.98	0.96	0.94	0.96
GPT4o-M	text	0.91	0.98	0.92	0.94	0.70	0.88	0.62	0.73	0.96	0.95	0.93	0.94
	image	0.75	0.91	0.71	0.79	0.91	0.95	0.85	0.90	0.86	0.94	0.83	0.87
	both	0.90	0.97	0.86	0.91	0.73	0.87	0.67	0.76	0.93	0.94	0.87	0.91
InternVL2	text	0.80	0.36	0.86	0.67	0.86	0.17	0.94	0.66	0.90	0.32	0.95	0.72
	image	0.65	0.34	0.71	0.57	0.98	0.39	0.99	0.78	0.88	0.42	0.95	0.75
	both	0.85	0.35	0.85	0.68	0.96	0.25	0.99	0.73	0.94	0.40	0.98	0.77
MiniCPM	text	0.96	0.79	0.99	0.92	0.96	0.78	0.97	0.90	0.98	0.83	0.99	0.93
	image	0.48	0.40	0.61	0.50	0.96	0.82	0.96	0.91	0.73	0.61	0.84	0.73
	both	0.89	0.69	0.95	0.84	0.93	0.78	0.95	0.89	0.96	0.78	0.98	0.91
Phi-V	text	0.81	0.09	0.94	0.61	0.82	0.12	0.85	0.60	0.89	0.12	0.87	0.63
	image	0.58	0.20	0.73	0.50	0.91	0.32	0.94	0.72	0.82	0.24	0.75	0.61
	both	0.74	0.18	0.87	0.60	0.85	0.18	0.87	0.63	0.89	0.16	0.86	0.64
LLaVA1.5	text	0.76	0.99	0.98	0.91	0.66	1.00	0.95	0.87	0.50	0.98	0.97	0.82
	image	0.40	0.89	0.74	0.68	0.75	1.00	0.98	0.91	0.33	0.92	0.83	0.69
	both	0.64	0.99	0.94	0.86	0.65	1.00	0.95	0.87	0.42	0.98	0.94	0.78
LLaVA-1.6 (Llama)	text	0.69	0.99	0.51	0.73	0.48	1.00	0.24	0.57	0.71	0.99	0.52	0.74
	image	0.42	0.93	0.33	0.56	0.72	0.99	0.36	0.69	0.54	0.96	0.40	0.64
	both	0.65	0.98	0.43	0.68	0.58	0.99	0.29	0.62	0.71	0.99	0.50	0.73
LLaVA1.6 (Mistral)	text	0.89	0.65	0.89	0.81	0.88	0.78	0.90	0.85	0.95	0.71	0.95	0.87
	image	0.59	0.37	0.59	0.52	0.89	0.74	0.93	0.86	0.79	0.55	0.81	0.72
	both	0.82	0.57	0.84	0.74	0.90	0.76	0.93	0.86	0.93	0.67	0.93	0.84
LLaVA1.5 RLAIF	text	0.19	0.86	0.66	0.57	0.37	1.00	0.62	0.66	0.22	0.89	0.35	0.49
	image	0.28	0.92	0.53	0.58	0.84	1.00	0.95	0.93	0.49	0.95	0.66	0.70
	both	0.22	0.89	0.55	0.55	0.54	1.00	0.74	0.76	0.33	0.92	0.50	0.58

Table A.6: CAST self-consistency scores for the first three statements generated for each modality configuration.

Table A.7: Telicity precision, recall, F1 score and accuracy results on RTS

Metric	Value
Precision	0.76
Recall	0.74
F1 score	0.75
Accuracy	0.81

Table A.8: Examples of template-based temporal questions and their LLM paraphrases.

Dataset	**Original sentence**	**Llama2 paraphrase**	**GPT-4 paraphrase**
Assembly101-QA	What happened before the person attached the wheel?	What was the antecedent to the wheel attachment?	Before the wheel was attached, what event took place?
Assembly101-QA	What object did the person attach the cabin to after attaching the step?	Following the attachment of the step, what did the person connect the cabin to?	Once the step was in place, what did the person then secure to the cabin?
STAR	Which object did the person close after they sat on the bed?	When sitting on the bed, what did the person make sure to close or latch beforehand?	Upon sitting on the bed, what object did the person close?
NExT-QA	How does the man position himself before hitting?	What stance or posture does the man adopt before swinging his bat?	What is the man's posture before he makes a hit?
NExT-QA	What did the girl do before she blew out the candles?	What was the girl's pre-candle blowing activity?	Prior to blowing out the candles, what did the girl engage in?

Table A.9: English templates: _past_ is the past simple form, _past_perf_ is the past perfect form, _ing_ is the -ing form including the past continuous form.

Template	MCA	DCA	Condition	n_persons	Question
1a	t	a	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	What did the person in the video do after v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 the obj_2 ?
1b	t	a	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	What did the person in the video do after the other person person v_dca_past_1 the obj_1 adjunct conj v_dca_past_2 the obj_2 ?
2a1	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	What did the person in the video do after they v_dca_past_perf_1 the obj_1 adjunct conj v_dca_past_perf_2 the obj_2 ?
2a2	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	What did the person in the video do when they v_dca_past_perf_1 the obj_1 adjunct conj v_dca_past_perf_2 the obj_2 ?
2b1	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	What did the person in the video do after the other person v_dca_past_perf_1 the obj_1 adjunct conj v_dca_past_perf_2 the obj_2 ?
2b2	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	What did the person in the video do when the other person v_dca_past_perf_1 the obj_1 adjunct conj v_dca_past_perf_2 the obj_2 ?
3a	t	a	et_mca <= st_dca	1	What had the person in the video done before v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 the obj_2 ?
3b	t	a	et_mca <= st_dca	2	What had the person in the video done before the other person v_dca_past_1 the obj_1 adjunct conj v_dca_past_2 obj_2 ?
4a	t	t	et_mca <= st_dca	1	What had the person in the video done before they v_dca_past_1 the obj_1 adjunct conj v_dca_past_2 the obj_2 ?
4b	t	t	et_mca <= st_dca	2	What had the person in the video done before the other person v_dca_past_1 the obj_1 adjunct conj v_dca_past_2 the obj_2 ?
5a1	t	a	et_mca > st_dca & et_mca < et_dca	1	What did the person in the video do v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 the obj_2 ?
5a2	t	a	et_mca > st_dca & et_mca < et_dca	1	What did the person in the video do while v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 the obj_2 ?
5b1	t	a	et_mca > st_dca & et_mca < et_dca	2	What did the person in the video do while the other person was v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 obj_2 ?
5b2	t	a	et_mca > st_dca & et_mca < et_dca	2	What did the person in the video do when the other person was v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 the obj_2 ?
6a	a	a	et_mca <= st_dca	1	What had the person in the video been doing before v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 the obj_2 ?
6b	a	a	et_mca <= st_dca	2	What had the person in the video been doing before the other person was v_dca_ing_1 the obj_1 adjunct conj v_dca_past_2 the obj_2 ?
7a	a	t	et_mca <= st_dca	1	What had the person in the video been doing before v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 the obj_2 ?
7b	a	t	et_mca <= st_dca	2	What had the person in the video been doing before the other person v_dca_past_1 the obj_1 adjunct conj v_dca_past_2 the obj_2 ?
8a1	a	a	st_mca > st_dca & st_mca < et_dca; st_mca == st_dca; st_mca < st_dca & et_mca > st_dca	1	What was the person in the video doing while v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 the obj_2 ?
8a2	a	a	st_mca > st_dca & st_mca < et_dca; st_mca == st_dca; st_mca < st_dca & et_mca > st_dca	1	What was the person in the video doing when v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 the obj_2 ?
8b1	a	a	st_mca > st_dca & st_mca < et_dca; st_mca == st_dca; st_mca < st_dca & et_mca > st_dca	2	What was the person in the video doing while the other person was v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 the obj_2 ?
8b2	a	a	st_mca > st_dca & st_mca < et_dca; st_mca == st_dca; st_mca < st_dca & et_mca > st_dca	2	What was the person in the video doing when the other person was v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 the obj_2 ?
9a	a	t	st_mca < et_dca & et_mca > et_dca	1	What was the person in the video doing when they v_dca_past_1 the obj_1 adjunct conj v_dca_past_2 the obj_2 ?
9b	a	t	st_mca < et_dca & et_mca > et_dca	2	What was the person in the video doing when the other person v_dca_past_1 the obj_1 adjunct conj v_dca_past_2 the obj_2 ?
10a	a	a	st_mca >= et_dca	1	What was the person in the video doing after v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 obj_2 ?
10b	a	a	st_mca >= et_dca	1	What was the person in the video doing after the other person v_dca_past_1 the obj_1 adjunct conj v_dca_past_2 the obj_2 ?
11a	a	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	What was the person in the video doing after v_dca_ing_1 the obj_1 adjunct conj v_dca_ing_2 the obj_2 ?
11b	a	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	What was the person in the video doing after the other person v_dca_past_1 the obj_1 adjunct conj v_dca_past_2 the obj_2 ?
12a1	t	t	et_mca == et_dca	1	What did the person in the video do when they v_dca_past_1 the obj_1 adjunct conj v_dca_past_2 the obj_2 ?
12a2	t	t	et_mca == et_dca	1	What did the person in the video do the moment they v_dca_past_1 the obj_1 adjunct conj v_dca_past_2 the obj_2 ?
12b1	t	t	et_mca == et_dca	2	What did the person in the video do when the other person v_dca_past_1 the obj_1 adjunct conj v_dca_past_2 the obj_2 ?
12b2	t	t	et_mca == et_dca	2	What did the person in the video do the moment the other person v_dca_past_1 the obj_1 adjunct conj v_dca_past_2 the obj_2 ?

Table A.10: Italian templates: _inf_ is the infinitive form, _pass_pross_ is the passato prossimo (present perfect) form, _trapass_pross_ is the trapassato prossimo (past perfect) form, _pass_rem_ is the passato remoto (simple past) form, _imp_ is the imperfetto (imperfect) form, _ger_ is the gerundio (gerund) form, _imp_cong_ is the imperfetto congiuntivo (imperfect subjunctive) form, _imp_prog_ is the imperfetto progressivo (past continuous) form. Conj is a coordinating conjunction in case of several verbs in one class, as in *lavorando o giocando al computer*.

Template	MCA	DCA	Condition	n_persons	Question
1a	t	a	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	Cosa ha fatto la persona nel video dopo v_inf_pass_pross_1 obj_1 adjunct conj v_inf_pass_pross_2 obj_2 ?
1b	t	a	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	Cosa ha fatto la persona nel video dopo che l'altra persona v_trapass_pross_1 obj_1 adjunct conj v_trapass_pross_2 obj_2 ?
2a1	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	Cosa fece la persona nel video dopo v_inf_pass_pross_1 obj_1 adjunct conj v_inf_pass_pross_2 obj_2 ?
2a2	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	Cosa fece la persona nel video quando v_pass_rem_1 obj_1 adjunct conj v_pass_rem_2 obj_2 ?
2b1	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	Cosa fece la persona nel video dopo che l'altra persona v_pass_rem_1 obj_1 adjunct conj v_pass_rem_2 obj_2 ?
2b2	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	Cosa fece la persona nel video quando l'altra persona v_pass_rem_1 obj_1 adjunct conj v_pass_rem_2 obj_2 ?
3a	t	a	et_mca <= st_dca	1	Cosa aveva fatto la persona nel video prima di v_inf_1 obj_1 adjunct conj v_inf_2 obj_2 ?
3b	t	a	et_mca <= st_dca	2	Cosa aveva fatto la persona nel video prima che l'altra persona v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
4a	t	t	et_mca <= st_dca	1	Cosa aveva fatto la persona nel video prima di v_inf_1 obj_1 adjunct conj v_inf_2 obj_2 ?
4b	t	t	et_mca <= st_dca	2	Cosa aveva fatto la persona nel video prima che l'altra persona v_trapass_cong_1 obj_1 adjunct conj v_trapass_cong_2 obj_2 ?
5a1	t	a	et_mca > st_dca & et_mca < et_dca	1	Cosa ha fatto la persona nel video v_ger_1 obj_1 adjunct conj v_ger_2 obj_2 ?
5a2	t	a	et_mca > st_dca & et_mca < et_dca	1	Cosa ha fatto la persona nel video mentre v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
5b1	t	a	et_mca > st_dca & et_mca < et_dca	2	Cosa ha fatto la persona nel video mentre l'altra persona v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
5b2	t	a	et_mca > st_dca & et_mca < et_dca	2	Cosa ha fatto la persona nel video quando l'altra persona v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
6a	a	a	et_mca <= st_dca	1	Cosa stava facendo la persona nel video prima di v_inf_1 obj_1 adjunct conj v_inf_2 obj_2 ?
6b	a	a	et_mca <= st_dca	2	Cosa stava facendo la persona nel video prima che l'altra persona v_imp_cong_1 obj_1 adjunct conj v_imp_cong_2 obj_2 ?
7a	a	t	et_mca <= st_dca	1	Cosa stava facendo la persona nel video prima di v_inf_1 obj_1 adjunct conj v_inf_2 obj_2 ?
7b	a	t	et_mca <= st_dca	2	Cosa stava facendo la persona nel video prima che l'altra persona v_imp_cong_1 obj_1 adjunct conj v_inf_2 obj_2 ?
8a1	a	a	st_mca > st_dca & st_mca < et_dca; st_mca == st_dca; st_mca < st_dca & et_mca > st_dca	1	Cosa faceva la persona nel video mentre v_imp_prog_1 obj_1 adjunct conj v_imp_prog_2 obj_2 ?
8a2	a	a	st_mca > st_dca & st_mca < et_dca; st_mca == st_dca; st_mca < st_dca & et_mca > st_dca	1	Cosa faceva la persona nel video quando v_imp_prog_1 obj_1 adjunct conj v_imp_prog_2 obj_2 ?
8b1	a	a	st_mca > st_dca & st_mca < et_dca; st_mca == st_dca; st_mca < st_dca & et_mca > st_dca	2	Cosa faceva la persona nel video mentre l'altra persona v_imp_prog_1 obj_1 adjunct conj v_imp_prog_2 obj_2 ?
8b2	a	a	st_mca > st_dca & st_mca < et_dca; st_mca == st_dca; st_mca < st_dca & et_mca > st_dca	2	Cosa faceva la persona nel video quando l'altra persona v_imp_prog_1 obj_1 adjunct conj v_imp_prog_2 obj_2 ?
9a	a	t	st_mca < et_dca & et_mca > et_dca	1	Cosa faceva la persona nel video quando v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
9b	a	t	st_mca < et_dca & et_mca > et_dca	2	Cosa faceva la persona nel video quando l'altra persona v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
10a	a	a	st_mca >= et_dca	1	Cosa faceva la persona nel video dopo v_inf_pass_pross_1 obj_1 adjunct conj v_inf_pass_pross_2 obj_2 ?
10b	a	a	st_mca >= et_dca	2	Cosa faceva la persona nel video dopo che l'altra persona v_trapass_pross_1 obj_1 adjunct conj v_trapass_pross_2 obj_2 ?
12a1	t	t	et_mca == et_dca	1	Cosa fece la persona nel video quando v_pass_rem_1 obj_1 adjunct conj v_pass_rem_2 obj_2 ?
12a2	t	t	et_mca == et_dca	1	Cosa fece la persona nel video nel momento in cui v_pass_rem_1 obj_1 adjunct conj v_pass_rem_2 obj_2 ?
12b1	t	t	et_mca == et_dca	2	Cosa fece la persona nel video quando l'altra persona v_pass_rem_1 obj_1 adjunct conj v_pass_rem_2 obj_2 ?
12b2	t	t	et_mca == et_dca	2	Cosa fece la persona nel video nel momento in cui l'altra persona v_pass_rem_1 obj_1 adjunct conj v_pass_rem_2 obj_2 ?

Table A.11: Russian Templates: _imp_ is imperfective past, _perf_ —perfective past. Conj is a coordinating conjunction in case of several verbs in one class, as in *читал или писал*.

Template	MCA	DCA	Condition	n_persons	Question
1a	t	a	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	Что сделал человек на видео после того, как v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
1b	t	a	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	Что сделал человек на видео после того, как другой человек v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
2a1	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	Что сделал человек на видео после того, как v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
2a2	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	Что сделал человек на видео, когда v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
2b1	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	Что сделал человек на видео после того, как другой человек v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
2b2	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	Что сделал человек на видео, когда другой человек v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
3a	t	a	et_mca <= st_dca	1	Что сделал человек на видео до того, как v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
3b	t	a	et_mca <= st_dca	2	Что сделал человек на видео перед тем, как другой человек v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
4a	t	t	et_mca <= st_dca	1	Что сделал человек на видео до того, как v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
4b	t	t	et_mca <= st_dca	2	Что сделал человек на видео перед тем, как другой человек v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
5a1	t	a	et_mca > st_dca & et_mca < et_dca	1	Что сделал человек на видео, пока он v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
5a2	t	a	et_mca > st_dca & et_mca < et_dca	1	Что сделал человек на видео, когда v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
5b1	t	a	et_mca > st_dca & et_mca < et_dca	2	Что сделал человек на видео, пока другой человек v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
5b2	t	a	et_mca > st_dca & et_mca < et_dca	2	Что сделал человек на видео, когда другой человек v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
6a	a	a	et_mca <= st_dca	1	Что делал человек на видео до того, как он v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
6b	a	a	et_mca <= st_dca	2	Что делал человек на видео до того, как другой человек v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
7a	a	t	et_mca <= st_dca	1	Что делал человек на видео до того, как он v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
7b	a	t	et_mca <= st_dca	2	Что делал человек на видео до того, как другой человек v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
8a1	a	a	st_mca > st_dca & st_mca < et_dca; st_mca == st_dca; st_mca < st_dca & et_mca > st_dca	1	Что делал человек на видео, пока v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
8a2	a	a	st_mca > st_dca & st_mca < et_dca; st_mca == st_dca; st_mca < st_dca & et_mca > st_dca	1	Что делал человек на видео, когда v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
8b1	a	a	st_mca > st_dca & st_mca < et_dca; st_mca == st_dca; st_mca < st_dca & et_mca > st_dca	2	Что делал человек на видео, пока другой человек v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
8b2	a	a	st_mca > st_dca & st_mca < et_dca; st_mca == st_dca; st_mca < st_dca & et_mca > st_dca	2	Что делал человек на видео, когда другой человек v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
9a	a	t	st_mca < et_dca & et_mca > et_dca	1	Что делал человек на видео, когда он v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
9b	a	t	st_mca < et_dca & et_mca > et_dca	2	Что делал человек на видео, когда другой человек v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
10a	a	a	st_mca >= et_dca	1	Что делал человек на видео после того, как он v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
10b	a	a	st_mca >= et_dca	2	Что делал человек на видео после того, как другой человек v_imp_1 obj_1 adjunct conj v_imp_2 obj_2 ?
11a	a	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	Что делал человек на видео после того, как он v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
11b	a	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	Что делал человек на видео после того, как другой человек v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
12a1	t	t	et_mca == et_dca	1	Что сделал человек на видео, когда v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
12a2	t	t	et_mca == et_dca	1	Что сделал человек на видео в тот момент, когда v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
12b1	t	t	et_mca == et_dca	2	Что сделал человек на видео, когда другой человек v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?
12b2	t	t	et_mca == et_dca	2	Что сделал человек на видео в тот момент, когда другой человек v_perf_1 obj_1 adjunct conj v_perf_2 obj_2 ?

Table A.12: Japanese templates: ta_form is the general past form, inf is the dictionary form, te_form is the conjunction form, i_form is the present progressive, and imp is the imperfect. Conj is a coordinating conjunction in case of several verbs in one class, as in ノートパソコンで仕事をしていたか、または遊んでいた.

Template	MCA	DCA	Condition	n_persons	Question
1a	t	a	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	ビデオに写っている人は adj obj を ta_form 後、何をしましたか？
1b	t	a	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	ビデオに写っている人は、他の人が adj obj を ta_form 後、何をしましたか？
2a1	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	ビデオに写っている人は adj obj を ta_form 後、何をしましたか？
2a2	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	ビデオに写っている人は adj obj を ta_form 時、何をしましたか？
2b1	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	ビデオに写っている人は、他の人が adj obj を ta_form 後、何をしましたか？
2b2	t	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	ビデオに写っている人は、他の人が adj obj を ta_form 時、何をしましたか？
3a	t	a	et_mca <= st_dca	1	ビデオに写っている人は adj obj を inf 前に、何をしましたか？
3b	t	a	et_mca <= st_dca	2	ビデオに写っている人は、他の人が adj obj を inf 前に、何をしましたか？
4a	t	t	et_mca <= st_dca	1	ビデオに写っている人は adj obj を inf 前に、何をしましたか？
4b	t	t	et_mca <= st_dca	2	ビデオに写っている人は、他の人が adj obj を inf 前に、何をしましたか？
5a1	t	a	et_mca > st_dca & et_mca < et_dca	1	ビデオに写っている人は adj obj を te_form 、何をしましたか？
5a2	t	a	et_mca > st_dca & et_mca < et_dca	1	ビデオに写っている人は adj obj を i_form ながら、何をしましたか？
5b1	t	a	et_mca > st_dca & et_mca < et_dca	2	ビデオに写っている人は、他の人が adj obj を inf_imp 間に、何をしましたか？
5b2	t	a	et_mca > st_dca & et_mca < et_dca	2	ビデオに写っている人は、他の人が adj obj を ta_form 時、何をしましたか？
6a	a	a	et_mca <= st_dca	1	ビデオに写っている人は adj obj を inf 前に、何をしていましたか？
6b	a	a	et_mca <= st_dca	2	ビデオに写っている人は、他の人が adj obj を inf_imp 前に、何をしていましたか？
7a	a	t	et_mca <= st_dca	1	ビデオに写っている人は adj obj を inf 前に、何をしていましたか？
7b	a	t	et_mca <= st_dca	2	ビデオに写っている人は、他の人が adj obj を inf 前に、何をしていましたか？
8a1	a	a	st_mca > st_dca & st_mca < et_dca; st_mca = st_dca; st_mca < st_dca & et_mca > st_dca	1	ビデオに写っている人は adj obj を i_form ながら、何をしていましたか？
8a2	a	a	st_mca > st_dca & st_mca < et_dca; st_mca = st_dca; st_mca < st_dca & et_mca > st_dca	1	ビデオに写っている人は adj obj を ta_form 時、何をしていましたか？
8b1	a	a	st_mca > st_dca & st_mca < et_dca; st_mca = st_dca; st_mca < st_dca & et_mca > st_dca	2	ビデオに写っている人は、他の人が adj obj を inf_imp 間に、何をしていましたか？
8b2	a	a	st_mca > st_dca & st_mca < et_dca; st_mca = st_dca; st_mca < st_dca & et_mca > st_dca	2	ビデオに写っている人は、他の人が adj obj を ta_form 時、何をしていましたか？
9a	a	t	st_mca < et_dca & et_mca > et_dca	1	ビデオに写っている人は adj obj を ta_form 時、何をしていましたか？
9b	a	t	st_mca < et_dca & et_mca > et_dca	2	ビデオに写っている人は、他の人が adj obj を ta_form 時、何をしていましたか？
10a	a	a	st_mca >= et_dca	1	ビデオに写っている人は adj obj を imp 後、何をしていましたか？
10b	a	a	st_mca >= et_dca	2	ビデオに写っている人は、他の人が adj obj を imp 後、何をしていましたか？
11a	a	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	1	ビデオに写っている人は adj obj を ta_form 後、何をしていましたか？
11b	a	t	st_mca >= et_dca; st_mca < et_dca & et_mca > et_dca	2	ビデオに写っている人は、他の人が adj obj を ta_form 後、何をしていましたか？
12a1	t	t	et_mca = et_dca	1	ビデオに写っている人は adj obj を ta_form 時、何をしましたか？
12a2	t	t	et_mca = et_dca	1	ビデオに写っている人は adj obj を ta_form 瞬間、何をしましたか？
12b1	t	t	et_mca = et_dca	2	ビデオに写っている人は、他の人が adj obj を ta_form 時、何をしましたか？
12b2	t	t	et_mca = et_dca	2	ビデオに写っている人は、他の人が adj obj を ta_form 瞬間、何をしましたか？

Table A.13: Model configurations and references.

Model	**Vision Encoder**	**Language Model**	**Parameters**
LLaVA-NeXT-Video-7B	SigLIP-400M [144]	Qwen 1.5 [7]	7B
MiniCPM-V-2_6	SigLIP-400M [144]	Qwen2 [7]	7B
Qwen2-VL	Qwen2-VL [129]	Qwen2 [7]	7B
InternVL2	InternViT-300M-448px [20]	internlm2_5-7b-chat [20]	8B

Appendix B

Appendix to PIE-V: Rubric Details, Agreement Tables, and Prompt Templates

This appendix provides supporting material for Chapter 5: (1) rubric wording and annotation guidance, (2) agreement and statistics tables, and (3) prompt templates and concrete examples used in PIE-V.

B.1 Procedural video datasets used throughout the thesis

In this section, we review the main procedural video datasets used throughout the thesis, both with and without errors. Beyond a high-level comparison (Table B.1), we describe in more detail the datasets that are central to this thesis: **EgoPER, EgoOops, Assembly101, CaptainCook4D, and Ego-Exo4D keysteps**. Ego-Exo4D keysteps provide clean, richly described source procedures for PIE-V. EgoPER, EgoOops, Assembly101, and CaptainCook4D serve as real-data references for mistake-aware procedural modeling. In Chapter 6, CaptainCook4D, EgoPER, and EgoOops are used in the main unified PCMD evaluation, while Assembly101 is retained as an auxiliary transfer benchmark from the earlier HMM-based study.

Most procedural datasets were created primarily for tasks such as action recognition, action segmentation, key step (sequence) extraction, object interaction, or pose estimation based on visual data. Consequently, instead of a full description of the steps, the annotations may take the form of action labels. This is typical of early datasets such as MECCANO [103], Assembly101 [111], ATA [41], and IndustREAL [107]. A shorter description of the procedure step makes it harder to recognize errors based on semantic cues.

Among error-aware datasets and tasks, the error annotations are heterogeneous and fragmented. The first group of general error annotations treats mistakes purely at the level of

sequence validity, i.e., whether the overall execution follows the canonical procedure, without localizing or typologizing individual erroneous steps. In CSV [101] each video of an experiment is labeled as correct or incorrect with respect to the entire reference protocol. Keystep-annotated parts of Ego-Exo4D [44] and ATA [41] focus on detecting whether an activity sequence adheres to the expected order, emphasizing structural deviations such as deletions and transpositions. The second group extends step/action annotations with per-step binary correctness labels. In HoloAssist [131], action segments are labeled as "correct" or "mistake", and conversational interventions are categorized (e.g., corrections, follow-ups), but the error label itself does not distinguish between procedural and executional issues. Assembly101 [111]-based benchmarks used in Ding et al. [26] attach a binary mistake flag to specific action segments, further distinguishing structural errors such as misordering or redundant steps, along with incorrect attachment of parts. Notably, this benchmark also marks corrective steps (detaching incorrectly attached parts) with a special label.

A smaller number of datasets introduce explicit taxonomies of both structural and execution errors, often tied to a specific domain. EgoPER [61] defines five error types assigned at the step level (omission, addition, modification, slip, correction). CaptainCook4D [96] provides a cooking-specific taxonomy (measurement, timing, temperature, technique, missing and misordered steps). EgoOops [47] adopts another multiclass execution-error taxonomy (incorrect-object-use, mispick, insufficient/excessive operation, wrong way of performing the step, accidents, and correction). CaptainCook4D and EgoOops augment each erroneous segment with a natural-language explanation of the error aligned to the procedural text (EgoOops additionally captures correction behavior in free-text explanations). Despite the richer label space, each taxonomy is closely tailored to a particular dataset.

In summary, Chapter 5 uses Ego-Exo4D keysteps as clean source procedures for PIE-V and uses EgoPER, EgoOops, Assembly101, and CaptainCook4D as real-data anchors; Chapter 6 uses CaptainCook4D, EgoPER, and EgoOops in the main unified evaluation and retains Assembly101 as an auxiliary benchmark for transfer and grounding analysis.

Assembly101. In the Assembly101 annotations, each step is represented by one action class and two object classes that are being manipulated. There are only two actions: *attach* and *detach*[2]. The full object vocabulary contains 64 parts, and some of them are semantically close (for example, "roller arm", "crane arm", and "excavator arm" can all be seen as instances of a more general "arm").

The classes only record the action and the objects, so if an object is attached incorrectly

[2]In the original annotations there is a third rare verb class, "position", which appears only together with the object "figurine". For the sake of simplicity it was merged into "attach".

(with a wrong orientation), this can be seen only from the error-type label "wrong orientation". To make the distinction clear at the text level, we converted class labels into full imperative commands using templates, for example, "attach the step to the chassis in the wrong way" versus the correct "attach the step to the chassis". In the original annotations, there is also no consistency between attaching part X to part Y and attaching part Y to part X. We normalized such steps to a single canonical form. As a result, we obtain a vocabulary of 339 full step descriptions.

Some toy variants may have only a single assembly in the whole dataset (e.g., `c13c` for a single correct assembly and `b04b` for a single erroneous assembly). Since the toy subtypes encoded by the last letter in the `toy_id` differ only slightly, we merge them into shared type classes to obtain more stable HMM training.

Overall, the error annotations in Assembly101 do not fully match our taxonomy, because the original annotations follow the logic of the assembly process rather than the logic of conformity to a reference procedure, as in our formulation. For example, from the assembly point of view, detaching an incorrectly attached part can be a correct step, but from the point of view of matching the canonical assembly, no detachment should be considered a correct step. Similarly, re-attaching a part after an erroneous detachment may be labeled as a *Correction*, but with respect to the reference procedure this step is simply correct. We therefore adjusted the annotations to match our task definition. The changes make it impossible to compare our model directly with other models that use Assembly101.

From the visual point of view, actions in Assembly101 are mirror-like: a detachment is the reverse of an attachment of the same parts. The parts in the dataset are also visually specific: they are sometimes small and visually similar to each other. In addition, the visual referent of the same object can change as the assembly progresses. For example, in assembly `c03f`, at the step "attach the arm connector to the chassis" the chassis has one appearance, while in the next step, "attach the body to the chassis", the term "chassis" refers to two already connected parts, the arm connector and the chassis. Thus, to recognize a step correctly, video models must not only capture the temporal dynamics of the action (which is a known weakness of many current VLMs, especially open-source ones), but also discriminate well between the objects involved in the step. In our experience, VLMs are not familiar with these parts, and the small number of videos, given this visual variability, makes fine-tuning particularly difficult.

CaptainCook4D. A key property of the cooking domain is the need to follow precise quantities to execute a recipe successfully. However, for visual models it is hard to see the difference between, say, "Add 1/3 tsp salt to the pan" (step_id: 178) and "Add 1/2 tsp salt to the pan" (step_id: 146), and even for a human this is often unclear from a single video. The dataset

authors also note in Peddi et al. [96] that full recipe understanding is multimodal rather than purely visual.

Even so, some dataset steps have very similar textual descriptions but different step IDs. For example, "Take 1 tomato" (step_id: 149) versus "Take a tomato" (step_id: 247), or "Peel 1 garlic cloves" (step_id: 200) versus "Peel 1 garlic clove" (step_id: 14). Such steps are visually indistinguishable. For each erroneous step, the modified textual description also provides the correction. However, it is not consistent. For example, in recording 1_33, the first step "Coat a 6-oz. ramekin cup with cooking spray" is labeled with the preparation error "Coating a large bowl instead of 6-oz ramekin cup", yet later steps are marked as correct and described as "Microwave the ramekin cup uncovered on high for 30 seconds", "Stir the ramekin cup" and so on. In the video the same bowl appears in all steps. Such inconsistencies in the step descriptions create discrepancies between the modalities.

Some steps also share almost the same temporal segment, but receive different textual descriptions and different step IDs. These steps are easy to distinguish in text but almost indistinguishable visually, adding further misalignment between the visual and textual modalities.

To be able to compare the performance of the models in the original CaptainCook4D paper with the performance of our model, we keep both the splits and the annotations exactly as released by the authors.

Given that each dataset contains less than 400 videos, these numerous inconsistencies contribute a substantial amount of noise. The small number of videos also makes it harder to fit robust models, such as HMMs.

EgoPER. EgoPER [61] is an egocentric cooking dataset built around a small set of recurring recipes (coffee, quesadilla, pinwheels, tea, oatmeal). It provides step-level timestamps and a psychologically motivated taxonomy with five labels: two structural deviations (step *omission* and step *addition*), two execution-level deviations (*slip* and *modification*), and *correction*. A distinctive aspect of EgoPER is that the annotations separate task-relevant steps from background activity: some segments reflect incidental actions that are not part of the core procedure (e.g., reading a script on a screen), and are marked explicitly as background rather than being forced into the step taxonomy. This design is helpful for studying mistake detection without conflating procedural steps with incidental context, but it also introduces a dataset-specific notion of "non-essential" activity that needs to be handled explicitly by models.

In Chapter 5, EgoPER is used as one of the key references for what human-like mistakes and corrections look like in real videos, and it informs our rubric-based evaluation of mistake

realism and procedure-level coherence.

EgoOops. EgoOops [47] contains 5 tasks with 10 videos per task. While it was designed to include both mistake-free executions and scripted mistake executions, in practice additional small deviations also appear in the "correct" runs (e.g., extra grasping or redundant manipulation), which makes the boundary between benign variation and mistake-like behavior particularly salient.

EgoOops provides a multi-class taxonomy of deviations (including *Correction*) and aligns each execution to a canonical script. However, for error segments the text often describes the *deviation from the canonical step* rather than the exact action the person performed. For example, instead of restating the full intended instruction, the annotation may specify what was wrong relative to the canonical step. This can be beneficial for supervised error recognition, but it complicates purely text-based modeling: recovering the implied correct step may require broader context and/or video grounding. Additionally, some steps are semantically dense and contain multiple predicates (e.g., "pour ... then dip ... and squeeze ..."), which increases the structural complexity of the instruction and the space of plausible errors.

In Chapter 5, EgoOops serves as a high-quality reference for visually plausible mistakes and recovery, and motivates several rubric dimensions (e.g., human plausibility and video plausibility) as well as the explicit modeling of corrections.

Ego-Exo4D keysteps. Ego-Exo4D [44] is a large-scale egocentric/exocentric dataset with multiple benchmarks. For PIE-V (Chapter 5), we specifically rely on the keystep annotations because they provide (i) step timestamps and (ii) natural-language step descriptions suitable as inputs for controlled textual rewriting.

A practical property of the keystep annotations is that step structure can be hierarchical: a step may be a leaf or a parent over finer-grained children, and some steps are explicitly marked as non-essential (i.e., present in the video but not required for the canonical goal). In PIE-V we use all available steps when constructing clean source procedures, because this reduces the risk of deleting or transposing critical actions when injecting errors, and it preserves a faithful "what happened" trace. We leave for future work a more aggressive setting that injects mistakes only into essential keysteps.

B.2 Rubric: details and scales

This section summarizes the rubrics as they were used in the annotation UI. The metrics for annotators are given in Table B.2, and the metrics propagated to LLM-as-a-judge are given in

Table B.3.

B.3 Agreement and statistics tables

"YES Statistics" means metrics computed *only on those events where a majority of annotators agreed the deviation is a mistake* (Error Validity = Yes). The results are shown in Tables B.5 (full) and 5.6 (YES).

B.4 Prompts and model identifiers

This section lists the key prompt templates used in PIE-V. For reproducibility, model identifiers are included as used in code:

- Writer (OpenAI): `gpt-5.2`
- Writer (Qwen): `Qwen/Qwen2.5-32B-Instruct`
- Judge (OpenAI multimodal): `gpt-5.2`
- Judge (Qwen multimodal): `Qwen/Qwen3-VL-32B-Instruct`

B.4.1 Freeform baseline prompt (LLM-only, no simulator)

The freeform baseline prompt requests 1–3 mistakes without a plan and forces strict JSON output.

```
### ROLE: Procedure Editor for '{scenario}'
You will receive a step-by-step procedure.
Your task is to produce an edited final procedure that contains 1 to 3
    plausible human mistakes.
You may also add 0 to 2 explicit correction steps.

### ERROR TYPES (HIGH-LEVEL)
- wrong_execution (mod='we'): Keep the same general goal, but execute it
    slightly wrong ...
- substitution (mod='s'): Replace the step with a different plausible
    action caused by confusion ...
- insertion (mod='i'): Insert one extra plausible step (unnecessary
    repetition, extra cleaning, checking, etc.).
- deletion: Remove the step completely (do not mention it was skipped).
    Add it to 'del'.
```

```
- transposition: Swap two steps.

### HARD CONSTRAINTS
1) VERBATIM PRESERVATION: Keep most steps unchanged unless directly edited.
2) IMPERATIVE STYLE: Use direct imperative commands. No story.
3) SOURCE INDEX RANGE: Every meta source_idx MUST be a valid original index.
4) PHYSICAL PLAUSIBILITY: Do not use objects before they appear earlier in the procedure.
5) METADATA ALIGNMENT: 'final_steps' and 'meta' must have the exact same length.
...
### OUTPUT FORMAT (STRICT JSON ONLY)
- final_steps: list[str]
- meta: list[list], one per final step: [source_idx, mod, error_id, correction_id]
- del: list[list] for deleted steps: [source_idx, error_id]
Return ONLY a single JSON object. No markdown. No extra text.
```

B.4.2 PIE-V writer prompt (plan-conditioned)

```
SYSTEM:
You are a careful procedure editor. You will rewrite a step-by-step procedure
according to an explicit ERROR-CORRECTION PLAN. You must:
- follow the plan exactly (error types, locations, corrections),
- preserve coherence and physical plausibility,
- keep most unchanged steps verbatim,
- propagate cascade edits when an earlier change affects later references.

You will be given:
(1) Original steps with indices
(2) Error events: type, target step, severity, semantic role edits (when applicable)
(3) Correction events: when to correct and how (redo / stop_and_fix / rollback_and_redo)

OUTPUT:
Return the final procedure as a JSON list of strings, in order.
Include correction steps as explicit imperative steps.
```

B.4.3 PIE-V judge prompt (coherence + plan compliance; optional multimodal)

```
SYSTEM:
You are a strict validator of a rewritten procedure containing planned
    mistakes.
Check:
- plan compliance (error type and placement),
- physical plausibility,
- logical coherence of the whole procedure,
- whether corrections truly repair the mistake.

If the procedure fails, return a revised corrected version that still
    follows the plan.
If an image is provided (mid-step frame), ensure the step text remains
    compatible with it.
```

B.5 Video synthesis: physically grounded prompts and stitching settings

This section shows excerpts from the Kling video generation scripts demonstrating: (i) physically grounded prompts (hands, objects, spatial relations), (ii) state-matching between clips using the last frame of clip A as the first frame of clip B, (iii) slightly longer crossfade for the critical error→correction boundary.

B.5.1 Coffee example prompts (A: spill, B: wipe and redo)

```
# Clip A: wrong execution (spill)
prompt_a = (‘‘‘‘‘‘
    Egocentric head-mounted POV, wide-angle fisheye with strong vignette.
    Same kitchen countertop scene.
    Continue naturally from the first frame.
    The small cup with milk is on the countertop in front of you.
    Action: start pouring coffee into the cup, but make a realistic
    mistake: tilt the coffee container slightly too fast so the stream
    briefly misses the cup and spills a SMALL amount of coffee onto the
    countertop right next to the cup (make a small puddle or splash).
    Some coffee should still go into the cup. Then stop the spill and set
     the coffee container down.
```

```
    Keep the spill visible on the countertop.
    Realistic liquid behavior, no teleporting, no sudden camera jumps.
    Do NOT introduce new objects, do not rearrange the countertop,
    preserve lighting and existing items.
    No text overlays.
    ''''''
    )

# Clip B: correction (wipe and redo) — starts from end of A
prompt_b = (``````
    Egocentric head-mounted POV, wide-angle fisheye with strong vignette.
    Same countertop, continue immediately from the first frame: there is
    a small coffee spill next to the cup.
    Action: put the coffee container down if it is in hand.
    Grab the YELLOW cloth/towel that is already on the countertop to the
    right, wipe the small coffee puddle with a few realistic strokes until
     the surface looks mostly clean/dry.
    Then pick up the coffee container again and carefully finish pouring
    coffee into the cup with milk, without any further spilling.
    Maintain consistent objects and placements, realistic motion, no cuts
    .
    Do NOT introduce new objects, do not change the scene, preserve
    lighting.
    No text overlays.
    ''''''
    )
```

B.5.2 Stitching settings (state match + crossfade)

```
# State matching:
# mid_b64 = last frame of clip A
# first_frame for clip B is set to mid_b64

# Crossfade tuning:
# xfade_duration_other = 0.25s
# xfade_duration_ab (A->B critical boundary) = 0.35s

# Quality depends on feature_ref:
# providing a feature reference video improves clip fidelity.
```

Table B.1: Procedural video datasets. **#Steps** refers to the number of distinct action/step classes when reported; otherwise it is left as "–". For **Step annotations**, *step* refers to natural step descriptions, while *action label* refers to verb + object(s) phrases. *Timestamps* mark either time or frame stamps.

Dataset	#Videos	Duration [h]	#Tasks	#Steps	Domains	Step annotations	Mistakes	Source
HowTo100M [82]	∼1.2M	> 100k	23k	–	various, incl. cooking	ASR narrations	×	YouTube
YouCook2 [156]	2,000	176	89	–	cooking	steps + timestamps	×	YouTube
COIN [117]	11,827	476	180	46,354	various, incl. repair	steps + timestamps	×	YouTube
CrossTask [157]	∼4.7k	∼374	83	–	various, incl. cooking	weakly aligned steps	×	YouTube
MECCANO [103]	32	–	1	–	toy assembly	action labels + timestamps	×	controlled lab
EPIC-KITCHENS-100 [25]	∼700	100	–	> 90k	cooking, kitchen activities	action labels + timestamps	×	participant recordings
50 Salads [80]	50	∼6.4	1	17	cooking	action labels	×	controlled lab
Breakfast [55]	77	77	10	48	cooking	action labels	×	semi-controlled participant recordings
IKEA ASM [12]	371	∼35	4	33	assembly	action labels + timestamps	×	controlled lab
EgoProceL [11]	329	62	16	139	various, incl. cooking, assembly	steps + timestamps	×	semi-controlled participant recordings
HoloAssist [131]	350	166	20	414	AR-assisted manipulations, incl. assembly	summary, conversations, steps + timestamps	✓	controlled lab
Assembly101 [111]	362	167	101	202	toy assembly	action labels + timestamps	✓	controlled lab
CaptainCook4D [96]	384	94.5	24	352	cooking	steps + timestamps	✓	participant recordings
EgoOops [47]	50	6.8	5	46	lab-style experiments and controlled assembly tasks	step + timestamp	✓	controlled lab
Ego-Exo4D (keysteps) [44]	852	30	17	186	various, incl. cooking, repair	step + timestamp	✓[1]	participant recordings
EgoPER [61]	396	28	5	70	cooking	step + timestamp	✓	participant recordings
ATA [41]	141	24.8	3	15	toy assembly	action labels	✓	controlled lab
EPIC-Tent [50]	24	> 5.4	1	38	assembly	action labels + timestamps	✓	participant recordings
BRIO-TA [85]	75	2.9	1	23	toy assembly	action labels + timestamps	✓	controlled lab
CSV [101]	70	11.1	14	106	chemical experiments	action labels	✓	controlled lab
IndustReal [107]	84	5.8	2	75	toy assembly	steps + timestamps	✓	Industrial-like lab

Table B.2: Rubric dimensions used for dataset assessment (human-facing criteria).

Metric	**Values**	**Annotator instructions (short)**	**Idea / analogues**
Human Plausibility	Likert 1–5	"Imagine a real person following this instruction and making this mistake. How plausible is it that someone would make exactly this mistake in this situation?" 1 – not plausible at all; 5 – very plausible.	Measures how human-like the error is in context (text + video). Likert scales are standard for perceived quality / naturalness in NLG eval [4, 59].
Taxonomy Fit (Error Type)	$\in$ {Deletion, Insertion, Substitution, Transposition, Wrong execution}	"Based on the step and its context, which label best describes how this step is wrong?" Deletion = missing required step; Insertion = unnecessary extra step; Substitution = replaced by different step; Transposition = wrong order; Wrong execution = wrong local parameters.	Assigns each error to a structured edit-based taxonomy to compare distributions across datasets. Related to error taxonomies used in current datasets and the survey [6].
Confusability / Difficulty to Notice	Likert 1–5 (higher = harder)	"Imagine you are performing this task, following the instruction. How easy would it be to overlook this mistake?" 1 – very easy to notice; 5 – very easy to miss (very subtle).	Proxy for detectability / severity. Conceptually related to detection in FMEA and to detectability notions in Signal Detection Theory.
Procedure Logic	$y \in \{Yes, No\}$; confidence $q \in \{1, 2, 3\}$	"Considering only the text, does this mistake make the sequence logically inconsistent (missing preconditions, impossible goals, impossible object changes), while the execution continues? Answer Yes/No and rate confidence (1–3)."	Captures global textual inconsistency (text-only). Related to plan/sequence coherence checks and ordering-constraint violations in procedural reasoning.
Sequence Consistency Score	Likert 1–5 (higher = more consistent)	"Consider text-only procedure. To what extent does each step appear when its natural prerequisites are satisfied?" 1 = many steps too early/late; 3 = mixed; 5 = coherent order (allow harmless permutations).	Global ordering coherence beyond strict logical breakage; connects to temporal coherence and ordering constraints in procedural reasoning [18, 26, 31, 60, 110, 119].
State-Change Coherence (Human Gold)	$x \in \{Yes, No\}$	If Procedure Logic is YES, check whether the implied world state becomes inconsistent at any point (objects appear/disappear, containers change without action, impossible preconditions, etc.). If Procedure Logic is NO, then always NO.	[45, 145].
Video Plausibility	Likert 1–5	"Ignoring the text, focus only on the video. Does the mistaken action look like something a real person might naturally do?" 1 – staged/unnatural; 5 – very natural.	Distinguishes staged vs natural mistakes; relates to realism/naturalness criteria in text-to-video evaluation [68, 145].
Text–Video Grounding Consistency (episode-level)	Likert 1–5	"Consider the whole video segment and its textual procedure (including the mistake and subsequent steps). How well does the text match what you see?" 1 – persistent mismatch; 5 – very well aligned.	Episode-level text–video alignment, aligned with human protocols in text-to-video evaluation [45, 145].

Metric	**Values**	**Annotator / model instructions (short)**	**Idea / analogues**
State-Change Coherence (LLM approx)	Integer #violations / score	For each adjacent step pair $(N, N{+}1)$, prompt an LLM to summarize state after N and judge whether $N{+}1$ follows logically; calibrate against human state-coherence labels on a subset. Conceptually related to contradiction/coherence checks and state tracking, and to validating automatic QA-style metrics against human judgments.	Scalable proxy for state coherence; LLM-as-judge inspired by NLI-style consistency checks and automatic QA evaluation paradigms.
LLM-based Plausibility / Cognitive Cause	Plausibility 1–5; cause $\in$ {slip, lapse, rule-based, knowledge-based, other}	Prompt LLM with procedure + mistake: (1) rate plausibility; (2) classify cognitive error type per Reason/Norman taxonomy with a short explanation; aggregate distributions.	LLM-assisted extrapolation grounded in classic human error taxonomies [88, 105].
Text–Video Grounding (VLM approx)	$[0, 1]$ alignment score	Use a VLM to answer fine-grained questions about whether actions/objects in text appear in the video (ETVA-style QA); validate against human grounding scores.	Automatic text–video alignment proxies in QA-based evaluation [45, 145].

Table B.3: Additional metrics and scalable approximations used to complement the human rubric.

Dataset	Err.Valid	Human Pl.	Confus.	Proc.Logic	Seq.Cons.	State-Chg	Taxonomy Fit	Vid.Pl.	T–V Gr.
EgoPER	0.912	0.541	0.368	0.728	0.628	0.579	0.759	0.574	0.662
EgoOops	0.916	0.592	0.375	0.836	0.667	0.600	0.882	0.579	0.560
Assembly101	0.859	0.584	0.697	0.739	0.649	1.000	0.931	0.670	0.861
CaptainCook4D	0.694	0.758	0.847	0.621	0.542	0.584	0.791	0.550	0.488
Ego-Exo4D-Qwen (freeform)	0.568	0.539	0.417	0.579	0.543	0.643	0.820	–	–
Ego-Exo4D-GPT-5.2 (freeform)	0.701	0.424	0.341	0.593	0.652	0.547	0.752	–	–
Ego-Exo4D-Qwen-PJ (PIE-V+Qwen2.5, Qwen3-VL-judged)	0.958	0.483	0.358	0.631	0.706	0.601	0.905	–	–
Ego-Exo4D-GPT-5.2-PJ (PIE-V+GPT, judged)	0.913	0.489	0.387	0.672	0.619	0.696	0.803	0.630	0.930

Table B.4: Krippendorff's α agreement summary. "–" indicates values are unavailable.

Dataset	Err.Valid (Yes/No)	Human Pl.	Confus.	Proc.Logic (Yes/No)	Proc.Logic Conf.	Seq.Cons.	State-Chg (Yes/No)	Vid.Pl.	T–V Gr.
EgoPER	Yes (51%), No (49%)	2.67	1.88	Yes (26%), No (74%)	2.74	4.41	Yes (14%), No (86%)	3.22	3.42
EgoOops	Yes (72%), No (28%)	3.73	1.63	Yes (28%), No (72%)	2.68	4.50	Yes (10%), No (90%)	4.31	3.74
Assembly101	Yes (65%), No (35%)	3.69	2.09	Yes (12%), No (88%)	2.78	4.82	Yes (4%), No (96%)	4.05	3.22
CaptainCook4D	Yes (74%), No (26%)	3.71	2.32	Yes (12%), No (88%)	2.73	4.73	Yes (2%), No (98%)	3.42	3.63
Ego-Exo4D-Qwen (freeform)	Yes (57%), No (43%)	3.22	2.02	Yes (36%), No (64%)	2.63	4.20	Yes (27%), No (73%)	–	–
Ego-Exo4D-GPT-5.2 (freeform)	Yes (83%), No (17%)	3.69	2.02	Yes (25%), No (75%)	2.85	4.81	Yes (7%), No (93%)	–	–
Ego-Exo4D-Qwen-PJ (PIE-V+Qwen2.5, Qwen3-VL-judged)	Yes (71%), No (29%)	3.17	1.41	Yes (30%), No (70%)	2.82	4.48	Yes (10%), No (90%)	–	–
Ego-Exo4D-GPT-5.2-PJ (PIE-V+GPT, judged)	Yes (89%), No (11%)	3.83	1.82	Yes (6%), No (94%)	2.91	4.91	Yes (3%), No (97%)	3.95	4.76

Table B.5: Aggregate statistics table. "Proc.Logic Yes" denotes the fraction of procedures judged logically broken (lower is better). Proc.Logic Conf. is reported only when Proc.Logic = Yes.

Appendix C

Appendix to ChronoFix: Semantic Representations, Hyperparameters, Configurations, and Grounding Details

C.1 Semantic representation construction

Table C.2 summarizes the core prompt design and output schema for SRL.

C.2 HMM Hyperparameters

The full list of HMM hyperparameters at all stages and for all modalities, with explanations, is given in Table C.3.

C.3 Reported configurations

This section lists the encoder and HMM settings behind the highest-scoring configurations reported in the main paper. We separate four cases: (1) the family-level leaderboard in Table 6.1, (2) the representation comparison in Table 6.2, (3) the search for one shared grounded configuration across all three datasets, and (4) the sensitivity of that shared choice to benchmark-specific selection metrics. Unless noted otherwise, each row corresponds to one complete run selected by the criterion stated in the main paper, and all reported settings are taken from that same run.

C.3.1 Family-level leaderboard configurations

Table C.4 lists the configurations for the family-level leaderboard in Table 6.1. Because each family is selected independently by Any-F1, the encoder, dimensionality, normalization, emission type, and alignment mode may differ across rows.

C.3.2 Representation comparison configurations

Table C.5 lists configurations corresponding to the values reported in Table 6.2. For each dataset/source/representation cell, we identify a run in the aggregated results file that reproduces the printed Any-F1 value up to the rounding used in the main paper. When multiple runs match the same rounded value, we report one representative match.

Dataset	Type	ID	Step description	SRL representation
CC4D	modified	20	Microwave the plate, covered, on high for 5 minutes instead of 1.5 minutes	`MICROWAVE(Agent: you, Object: plate(State: covered), Setting: high, Duration: minute(Quantity: 5), Manner: INSTEAD_OF(Duration: minute(Quantity: 1.5)))`
	modified	28	Measure 1/8 teaspoon of salt using dirty measuring spoon and add it to the mug	`ADD(Agent: you, Object: salt(Measurement: teaspoon(Quantity: 1/8)), Destination: mug, Instrument: dirty(measuring_spoon), Manner: after(MEASURE(Agent: you, Object: salt(Measurement: teaspoon(Quantity: 1/8)), Instrument: dirty(measuring_spoon))))`
	canonical	54	Put all the Vegetables in a microwave-safe bowl	`PUT(Agent: you, Object: vegetables(Quantity: all), Destination: bowl(Property: microwave_safe))`
EgoOops	canonical	211	Put chopsticks through the cutout in the lid to widen the hole.	`PUT(Agent: you, Object: chopsticks, Path: through(cutout(Location: in(lid))), Purpose: WIDEN(Agent: you, Object: hole))`
	modified	10	Pour about 15mL of water into a cup, dip the tip of a red highlighter in the water and squeeze out a drop. Knock over an empty cup.	`POUR(Agent: you, Object: water(Measurement: ml(Quantity: 15, Approx: about)), Destination: into(cup)); DIP(Agent: you, Object: tip(of(red_highlighter)), Destination: in(water(Location: in(cup)))); SQUEEZE_OUT(Agent: you, Object: drop); KNOCK_OVER(Agent: you, Object: empty_cup)`
EgoPER	canonical	57	Put banana slices on tortilla	`PUT(Agent: you, Object: banana_slices, Destination: on(tortilla))`
	canonical	37	Microwave for X seconds	`MICROWAVE(Agent: you, Temporal: for(second(Quantity: x)))`
	canonical	18	Drop tea bag	`DROP(Agent: you, Object: tea_bag)`

Table C.1: Representative SRL examples from the three benchmarks. We include both canonical and modified step descriptions to show how the representation handles quantities, instruments, destinations, purposes, and multi-event modified steps. CC4D is CaptainCook4D.

Component	Content
System instruction	You are a linguistic semantic analyzer. For each procedural step description, generate the semantic representation in roles.
Prompt guidance	Predicates are uppercase. Concrete manipulated entities are usually encoded under `Object`. `Agent: you` is used unless another agent is explicitly specified. Entity names are normalized as lowercase_with_underscores. Nested structures are allowed when needed, e.g., `tip(of(red_highlighter))` or `water(Location: in(cup))`. The prompt also gives typical role names such as `Object`, `Coobject`, `Destination`, `Location`, `Instrument`, `Manner`, `Purpose`, `Temporal`, `Origin`, and `Result`.
Selected few-shot examples	`INSERT(Agent: you, Object: test_swab, Destination: into(nostril(of(her))))`; `ADD(Agent: you, Object: coffee_grounds, Origin: bowl, Destination: filter(Location: in(french_press)))`; `ADD(Agent: you, Object: cut(onions), Coobject: egg(Location: in(mixing_bowl)))`; `TOP(Agent: you, Object: cup, Coobject: with(salsa(Measurement: tablespoon(Quantity: 1))))`; `POUR(Agent: you, Object: egg(Quantity: 1), Destination: into(ramekin_cup))`.
Output schema	Strict JSON with one item per step, containing the fields `id`, `step_description`, and `semantic_representation`.
Post-processing	Manual edits were sparse and limited to targeted corrections of role assignment, attachment, and lexical normalization.
Covered vocabularies	Benchmark descriptions from correct videos, modified descriptions for erroneous steps, and VLM-predicted descriptions not already covered by the first two vocabularies.

Table C.2: Core design of the prompt and schema used to generate SRL representations.

Table C.3: The full list of the HMM's hyperparameters with default settings and explanations. Stages: training (**t**), inference (**i**), and multimodal (**mm**).

Hyperparameter	**Sym**	**Stg**	**Meaning**	**Value effect**	**Default settings & rationale**
Frames per step	n	t/i	*Number of frames* per step averaged into the image feature.	High: stronger denoising and object coverage; more IO/latency; can over-smooth very short steps. Low: faster, crisper but noisier; more sensitive to accidental frames.	**3**: good robustness–latency trade-off; raising adds to visual noise, but might be useful for for long, stable steps.
PCA dimension	d	t/i	Output dimensionality after PCA (fit on train; applied at inference) before Gaussian emissions.	High: preserves fine semantics; harder covariance estimation, overfitting. Low: better conditioning/sample-efficiency; too low drops discriminative cues.	**128**: stable Ledoit–Wolf covariances with modest memory/compute; increase with ample per-class data.
Laplace smoothing	β	t	Add-β smoothing for transition A and start prior π (counts+β, renormalize).	High: flatter A, π; safer on rare/omitted transitions; less brittle. Low: sharper priors mirroring train paths; risk overfitting ordering quirks.	**3.0**: pairs well with near-diagonal banding to encourage local moves without hard constraints.
Banding (local moves)	band	t	Mask A_{ij} if $\lvert i-j\rvert > 1$, then renormalize rows; encodes locally ordered procedures.	On: curbs long jumps/cycles; more interpretable paths. Off: allows global jumps for branched workflows; higher relocation FPs.	**On** with $\lvert i-j\rvert \leq 1$: disabling recommended only if tasks truly need non-local transitions.
Jitter probability	p_{jit}	t	Probability to add Gaussian noise to a training vector (after norm/PCA).	High: more invariance to ambience/lighting; too high blurs class boundaries. Low: crisper clusters; brittle to domain shift.	**0.15**: mild diversity without collapsing adjacent steps; co-tuned with σ.
Jitter std	σ	t	Noise scale in $\mathcal{N}(0, \sigma^2 I)$ (PCA space if used).	High: robust but can distort likelihoods and overlap classes. Low: minimal regularization.	**0.3**: keeps perturbations realistic relative to intra-class spread.
Per-modality normalization	l2/None	t/i	L2-normalize each modality before fusion and HMM; stabilizes scales and PCA.	l2: avoids dominance by high-variance channels; steadier thresholds. None: preserves raw magnitudes; can skew fusion and Gauss LL.	**l2**: matches CLIP best practice; lets w modulate content, not scale.
Emission family	Gauss/Cosine	i	Observation model at inference: Gaussian log-likelihoods vs. cosine similarity (with temperature).	Gauss: calibrated LL, leverages PCA+Ledoit–Wolf. Cosine: magnitude-invariant; needs temperature; transitions may need rebalancing.	**Gauss** default; switch to **Cosine** for strongly angular features or when skipping PCA.
Cosine temperature	T	i	Scale for cosine emissions ($E = T \cdot \cos$); controls emission/transition balance.	High: emission-dominant, peak-following paths. Low: transition-dominant, smoother paths.	**N/A** for Gauss. For Cosine, tuned T to match the dynamic range of log-transitions.
Emission scaling	α	i	Global multiplier $\tilde{E} = \alpha E$ to retune emission vs. transition influence.	High: emission-driven Viterbi (sensitive to per-step scores). Low: transition-driven (smoother, prior-driven).	**1.0**: adjustment needed only if domain shift skews score ranges.
Emission anomaly threshold	θ_e	i	Flag if $\log p(\mathbf{z}_t \mid S_t) < -\theta_e$ (low-prob emission).	High: ↑ recall, ↓ precision for LowProbEmission. Low: ↑ precision, ↓ recall.	**Auto-calibrated** from state LL stats (e.g., mean/quantiles) to target a desired FPR.
Transition anomaly threshold	θ_t	i	Flag if $\log A_{S_{t-1},S_t} < -\theta_t$ (low-prob jump).	High: flags rare jumps aggressively. Low: conservative; may miss subtle re-orderings.	**Auto-calibrated** from observed transition stats; align to tolerance for reordering.
Alignment mode	NW/difflib	i	Align prediction to a canonical variant to label Deletion/Insertion/Substitution/Relocation.	NW: stable edit metric (use sub=2, gap=1 for reports). difflib: forgiving for narratives and qualitative inspection.	**difflib**: generally performs better.
Global fallback HMM	task$=-1$	i	Task-agnostic model pooled over all tasks when per-task params are missing or weak.	On: graceful degradation on rare or unseen tasks. Off: stricter specialization; risk of empty or degenerate paths.	**On**: helps tail tasks.
Vision weight (mm)	w	mm (t/i)	Fusion in joint space: $\mathbf{x} = w\,\mathbf{v}_{img} + (1-w)\,\mathbf{v}_{txt}$ (or scaled-concat in disjoint space).	High: vision-led. Low: text-led; better when phrasing disambiguates or clarifies visuals.	**Task-dependent**. Optimal $w \approx 0.2$–0.4: for subtle manipulations; increase when distinctive visuals drive the step.
Use superstructure (mm)	on/off	mm (t/i)	Add structured phrases (objects/actions) as text before fusion.	On: boosts separability for visually similar steps; sensitive to noisy phrases. Off: simpler; may blur key discriminators.	**Off** for clean baselines; **On** in V + AO style variants with reliable extractors and prompt filtering.
Superstructure top-k	k	mm (t/i)	Keep top-k phrases by confidence to control context breadth.	High: broader context, more recall but more distractors. Low: focused cues; risk missing secondary hints.	$k = 3$–5: small slate gives gains without flooding the encoder.
Superstructure threshold	τ	mm (t/i)	Minimum confidence for phrases included in superstructure.	High: precise/sparse cues. Low: recall-oriented; may add noise and hurt calibration.	**Moderate** τ: to be tuned jointly with k for coverage vs. noise.

Dataset	Family	Encoder	d	Norm	Emission	Temp	Align
EgoOops	Vision	V-JEPA	1024	L2	Gaussian	–	NW
	Raw text	CLIP	128	none	cosine	6	NW
	SRL	CLIP	128	none	cosine	12	difflib
	Qwen	Instructor-XL	64	L2	Gaussian	–	difflib
	Qwen + SRL	Instructor-XL	64	none	cosine	–	difflib
	Gemini	all-MPNet	768	L2	Gaussian	–	difflib
	Gemini + SRL	Instructor-XL	64	none	cosine	–	difflib
EgoPER	Vision	VideoMAE-v2	128	L2	cosine	6	difflib
	Raw text	CLIP	256	none	Gaussian	–	difflib
	SRL	RoBERTa-SimCSE	256	L2	Gaussian	–	NW
	Qwen	CLIP	256	L2	Gaussian	–	NW
	Qwen + SRL	RoBERTa-SimCSE	256	none	Gaussian	–	NW
	Gemini	CLIP	256	L2	Gaussian	–	NW
	Gemini + SRL	RoBERTa-SimCSE	128	L2	Gaussian	–	NW
CaptainCook4D	Vision	VideoMAE-v2	768	none	cosine	6	difflib
	Raw text	RoBERTa-SimCSE	256	none	cosine	10	NW
	SRL	RoBERTa-SimCSE	256	none	cosine	10	difflib
	Qwen	all-MiniLM	256	L2	cosine	12	difflib
	Qwen + SRL	RoBERTa-SimCSE	256	L2	cosine	10	difflib
	Gemini	RoBERTa-SimCSE	128	none	cosine	12	difflib
	Gemini + SRL	RoBERTa-SimCSE	256	none	cosine	10	difflib

Table C.4: Configurations for the family-level leaderboard in Table 6.1. Each row lists the configuration attaining the highest Any-F1 within that representation family.

Dataset	Src.	Repr.	Encoder	d_{raw}	d	Norm	Emiss.	Temp	Align
EgoOops	Benchmark	Raw	OpenAI	256	256	L2	cosine	8	NW
		SRL	all-MPNet	–	128	L2	cosine	8	NW
		Act.+Obj.	OpenAI	256	256	L2	cosine	12	NW
	Qwen grounding	Raw	Instructor-XL	–	128	L2	cosine	8	NW
		SRL	Instructor-XL	–	128	L2	cosine	8	NW
		Act.+Obj.	OpenAI	256	256	L2	Gauss.	–	difflib
	Gemini grounding	Raw	Instructor-XL	–	128	L2	cosine	8	NW
		SRL	Instructor-XL	–	128	L2	cosine	8	NW
		Act.+Obj.	OpenAI	64	64	L2	Gauss.	–	difflib
EgoPER	Benchmark	Raw	OpenAI	768	768	L2	Gauss.	–	difflib
		SRL	OpenAI	768	768	L2	Gauss.	–	NW
		Act.+Obj.	OpenAI	128	128	none	Gauss.	–	NW
	Qwen grounding	Raw	RoBERTa-SimCSE	–	128	L2	Gauss.	–	NW
		SRL	RoBERTa-SimCSE	–	128	L2	Gauss.	–	NW
		Act.+Obj.	OpenAI	64	64	L2	Gauss.	–	NW
	Gemini grounding	Raw	RoBERTa-SimCSE	–	128	none	Gauss.	–	difflib
		SRL	RoBERTa-SimCSE	–	128	L2	Gauss.	–	NW
		Act.+Obj.	OpenAI	64	64	L2	Gauss.	–	NW
CC4D	Benchmark	Raw	RoBERTa-SimCSE	–	64	none	cosine	8	NW
		SRL	OpenAI	128	128	L2	cosine	8	difflib
		Act.+Obj.	OpenAI	768	128	L2	cosine	12	difflib
	Qwen grounding	Raw	RoBERTa-SimCSE	–	128	L2	cosine	10	difflib
		SRL	RoBERTa-SimCSE	–	256	L2	cosine	10	difflib
		Act.+Obj.	OpenAI	64	64	L2	cosine	6	difflib
	Gemini grounding	Raw	RoBERTa-SimCSE	–	256	L2	cosine	10	difflib
		SRL	RoBERTa-SimCSE	–	256	L2	cosine	10	difflib
		Act.+Obj.	OpenAI	128	64	none	cosine	12	NW

Table C.5: Representative configurations corresponding to the Any-F1 values reported in Table 6.2. Each row reproduces the printed value for that dataset/source/representation cell up to the rounding used in the main paper. CC4D denotes CaptainCook4D.

Method	Acc	P	R	F1	AUC
ImageBind V3 (V,A,T), Split R [96]	64.08	44.15	58.01	50.13	58.35
Ours (best grounded)	**66.15**	57.97	**70.80**	**63.75**	**75.48**
Ours (shared config)	65.78	**58.25**	65.63	61.72	74.55

Table C.6: Contextual comparison to the Split R `V,A,T` baseline from the original CaptainCook4D paper.

Dataset	Reference step	Observed / grounded variant	Why SRL helps
CC4D	Place the egg from the cup over the lettuce	Place the egg from the cup under the lettuce	Makes the relation contrast explicit: `over` vs. `under`.
	Invert the mug to release the cake onto a plate	Invert the mug to release the cake onto a bowl Invert the mug to release the cake onto an unclean plate	Isolates destination and destination-property changes while preserving the same event structure.
	Invert the mug to release the cake onto a plate	Use a spoon to invert the mug to release the cake onto a plate	Preserves the added instrumental subevent as nested structure.
	Measure and add 1.5 tbsp sugar to the mixing bowl	Qwen: Measure and add 1.5 tbsp brown sugar to the mixing bowl	Preserves the `MEASURE & ADD` structure and highlights the ingredient mismatch.
EgoOops	Put three zinc plates on the center column of the microplate using the pair of tweezers	Qwen: Put three copper plates on the left column of the microplate using the pair of tweezers	Makes both material and location drift explicit.

Table C.7: Qualitative examples where SRL is more informative than raw grounded text alone. The main benefit comes from making relation, destination, property, material, and nested action structure explicit. CC4D denotes CaptainCook4D.

C.4 Additional qualitative examples

This appendix illustrates the qualitative patterns behind the main quantitative results. We focus on three questions: when SRL helps over raw grounded text, what the action–object abstraction preserves and loses, and how the unified edit-based taxonomy applies across heterogeneous benchmarks.

C.4.1 When SRL helps

Table C.7 shows cases where SRL is more informative than raw grounded text alone. SRL is most useful when the mistake lies in a relation, destination, material, or nested event structure, or when a VLM drifts lexically while still producing a nearby step description.

C.4.2 What action–object preserves and loses

Table C.8 shows what the action–object abstraction preserves and what it weakens. Our action–object representation retains the main action together with all visually involved ob-

Effect	Dataset	Reference step	Contrasting step	What action–object captures or loses
Preserves	EgoPER	Use knife to scoop jelly	Use spoon to scoop jelly	Preserves the tool distinction because tools remain in the object set.
	CC4D	Sprinkle 1 tablespoon of cheese on cup	Pour 1 tablespoon of cheese on cup	Preserves the predicate change.
Weakens	CC4D	Microwave just until cheese melts, about 10 seconds	Microwave for 30 seconds	Loses duration and completion-condition information.
	CC4D	Top cup with 1 tablespoon of salsa	Top cup with 1/2 tablespoon of salsa	Loses scalar quantity information.
	EgoPER	Slowly pour the rest of water in circular motion	–	Loses manner information.
	CC4D	Replace the top of the English muffin	Flip over the top muffin and replace it on top	Weakens multi-action structure.

Table C.8: Qualitative examples of what the action–object abstraction preserves and what it weakens. It remains informative for predicate and tool changes, but loses detail when the mistake depends on duration, scalar quantity, manner, or multi-action structure. CC4D denotes CaptainCook4D.

jects, not only the semantic patient. Tools and contacted objects are therefore preserved, which makes the representation useful for some mistakes but too coarse for others.

C.5 Action and Object Superstructure Details

To enrich the **CaptainCook4D** dataset, we generated a new set of dense annotations focused on **action** and **object** labels for each procedural video step. These annotations capture both the primary activity and the full set of relevant entities involved.

Action Label Generation. The action label represents the core activity of a step. It was primarily derived by isolating the main verb from the step description.

- **Standard Case:** The action is directly the main verb (e.g., from "Peel 1 medium onion," the action is "peel").
- **Edge Case Handling:** To ensure semantic consistency and better reflect the visual reality of the video, we manually refined labels for steps where the main verb did not accurately represent the performed activity. For instance, for the description: "Let the

noodles sit for about 1 minute after the microwave stops," the action was labeled "`wait`," as this is the most representative and generalizable temporal action in this context.

Object Label Generation The object label is a comprehensive list of all entities (both ingredients and tools) that are either explicitly mentioned or actively involved in the execution of the step.

- **Parsing and Extension:** We first parsed the explicitly involved objects from the step description. Then we extended this initial list to include necessary tools that are commonly used to execute the action, even if they were not explicitly mentioned in the text but were visually present and crucial to the step's execution.
- **Example:** For the step description "Chop 1 garlic clove on a cutting board," the resulting object labels are "`garlic clove`", "`cutting board`", and "`knife`". The object "`knife`" is included because it is the most visible and essential tool for 'chop'.

The labeling strategy for the **Assembly101** dataset is tailored to its simpler, primarily manipulative task structure, where procedural steps focus on assembling toy parts. Since the procedural steps are dominated by a limited set of binary actions "attach" and "detach" and for the correct procedure videos only the "attach" action is relevant, the action label is redundant to distinguish between steps. Consequently, we do not use the action label for this dataset.

The object labels for Assembly101 are directly derived from the components mentioned in the step description. We extended this to track the state change of objects on the workspace.

- **Direct Parsing:** The primary objects involved in the manipulation were extracted directly from the step descriptions (e.g., from "attach the wheel to the chassis," the objects are "`wheel`" and "`chassis`").
- **Complementary Objects (Workspace State):** Since all toy parts begin scattered on a table and disappear as they are assembled, we created a list of complementary objects for each step. This list represents the full set of component parts that are still visible and available on the table at the beginning of the step. This provides a crucial visual context regarding the current state of the assembly and the components yet to be used. These toy parts remain highly visible across frames, in contrast to assembly components that are often occluded by hands.

In our experiments with VLMs, we observe that many recognition errors do not stem from failures in visual perception or object discrimination, but rather from the highly domain-specific nature of toy components. As a result, models often identify the object visually yet

fail to assign the correct canonical name. For example, GPT-5 describes one component as a "black-and-yellow attachment piece ⋯resembling a front attachment for a construction toy vehicle", whereas the correct label is simply "cabin". To mitigate this type of mistakes we created a comprehensive object dictionary. This dictionary is maintained as a supplementary CSV file, providing detailed visual and physical descriptions for unique or ambiguous object names. A sample of this dictionary is given in Table C.9.

Table C.9: Sample Entries from the Assembly101 Object Dictionary.

Object Name	Visual Description
`arm`	Yellow arm piece with a flat rectangular shape and a curved cutout at one end.
`arm connector`	Small yellow plastic arm connector with a rounded joint section and a flat rectangular body.
`cabin`	Plastic car body shell and a label on top — lightweight and hollow, shaped like the outer frame of a toy vehicle.
`fire equipment`	Small yellow plastic piece with two short pegs at the bottom and a rectangular frame shape; it has a gray cylindrical element housed inside the center. The outer frame is transparent yellow, and the gray section contrasts distinctly within it.
`tilter`	Yellow plastic piece with a long rectangular body, a round dark pivot in the center, and a hole at each end. This toy part is a small, yellow, symmetrical piece with a distinct "C" shape when viewed from the front.

C.5.1 Prompts for CaptainCook4D

In this subsection, we detail the prompt formulations used to elicit step descriptions, action labels, and object predictions for each video clip in the CaptainCook4D dataset. We first present the prompts employed for image-based Visual Language Models, specifically Gemma-3 and GPT-4o. We then describe the prompt designs used for Video Language Models: Gemini and InternVL-3.

VLM Prompt for Step Description Prediction. Figure C.1 shows the VLM prompt to analyze the input video frames (as visual input) along with contextual textual information (step number, activity name, and list of all possible steps) to predict the most likely procedural step. The variable `{to_add}` is not part of the static prompt. It adds task-specific warnings when potentially confusing objects appear in the procedure. For example:

- If "toothpicks" appear in all possible objects per task, the system appends: "Do not confuse butter with jelly and toothpicks."

```
You are given egocentric frames captured during the execution of a step
   number {step_number} of total {len(steps)} steps of procedure: {
   activity_name}.

LIST of all possible steps:
[{all_possible_steps_per_task}]

Analyze frames and objects involved (ignore the laptop screen and
   consider context of {activity_name} procedure) to understand the step
   the user is doing. Take into account a step number of step. Do not be
   biased and choose the most likely step and estimate probability.

{to_add}.

Answer only json format:
{
    ''task'': <procedure_name>,
    ''steps'': [
        {
            ''step_name'': <step_name>,
            ''probability'': <prob_value>
        }
    ]
}
```

Figure C.1: Prompt used to predict the step description.

- If "sugar" appears, the system appends: "Do not confuse sugar and flour."

VLM Prompt for Action Prediction. Predicting actions proves challenging because the dataset contains 77 distinct fine-grained actions. Using only the actions is insufficient, as VLMs struggle to distinguish subtle visual differences across the large action set. Alternative prompting strategies improves action prediction. One approach involves providing step-level textual context and asking the model to jointly predict both the step description and the action. Interestingly, the predicted action is not always consistent with the model' s own generated step. For example, for the ground-truth step "Spread nut butter onto the tortilla" GPT-4o predicts action "spread" (which is correct) but the predicted step is "Use a butter knife to scoop nut butter from the jar". Ultimately, the model performance improves when the list of task-relevant objects is included in the prompt context. Our final prompt (Figure C.2) instructs the model to analyze the visual frames together with the provided object list, identify the primary objects involved in the action (up to three), ignore irrelevant elements such as clothing or background artifacts, and output the most relevant action from the predefined action set.

VLM Prompt for Objects Prediction. We annotated each step with a subset of 231 unique objects in CaptainCook4D. As most video steps involve only three or fewer relevant objects, we ask the VLM to predict only the top three objects. In practice, this introduce a ranking

```
You are given egocentric frames captured during the execution of a step
    number {step_number} of total {len(steps)} steps of procedure: {
    activity_name}.

LIST of all possible objects:
[{all_possible_obj_per_task}]

LIST of all possible actions:
[{all_possible_act_per_task}]

Analyze frames and objects involved in the depicted action. Choose names
    of objects with probabilities and action name with probability. Do not
     pay attention to people clothes, the laptop screen or irrelevant for
    action objects. Only primary objects. If you see only one object
    involved, choose only it. But no more than 3 objects. Do not be biased
    . {to_add}. Write the most relevant action from the provided list.
    Answer only json format:
{
    ''task'': <procedure_name>,
    ''actions'': [
        {
            ''act_name'': <action_name>,
            ''prob'': <prob_value>
        }]
}
```

Figure C.2: Prompt used to predict the core action.

bias, and relevant objects are occasionally excluded from the top-3 set. To address this issue, we adopt a binary relevance-classification scheme in which the VLM independently judges each candidate object as relevant or not. Figure C.3 shows the prompt that iterates through a list of all possible objects for the task, asking a separate yes/no question for each.

Unified Prompt for Video-Native Models (Gemini-2.5-Flash, InternVL3-8B)

For Gemini-2.5-Flash and InternVL3-8B, we use a single multi-task prompt (Figure C.4), which leverages their temporal modeling capabilities.

C.5.2 Prompting for Assembly101

For Gemini-2.5-Flash and GPT-5 models, the prompt provides (i) candidate steps, (ii) relevant toy parts, and (iii) explicit instructions to ignore irrelevant elements and avoid common confusions between visually similar components.

We predict both **assembling objects** and **unassembled toy parts**. As most parts remain in the same positions across steps unless used, we thus verify object detection accuracy at each step. For evaluation, we created step-wise labels and ensure that assembling objects and

```
{to_add}

The questions dynamically generated for all possible objects per task:
for obj in all_possible_obj_per_task:
    questions += f''Is {obj} a main object to describe the action in the
        pictures?\n''

Answer only yes or no. Only json format:
{
    ''objects'': [
        {
            ''obj_name'': <obj_name>,
            ''answer'': <yes or no>
        }]
}
```

Figure C.3: Prompt used to predict the objects.

```
You are given a video segment captured during cooking.
LIST of all possible steps: [{all_possible_descriptions_str}]
Analyze the primary action and most relevant objects to determine the
   executed step.
Choose the most likely step and estimate its probability.
{to_add}.
Answer only json format:
{
    ''step_name'': <step_name>,
    ''action_name'': <action such as add, cut, etc.>,
    ''primary_objects'': ''<object1>, ...''
}
```

Figure C.4: Multi-task prompt

unassembled toy parts do not intersect.

We also explored alternative prompting strategies, including a few-shot setup in which the model first described, in its own words, all objects on the table (both assembled and unassembled) and then attempted to match these descriptions to the set of possible toy parts. This approach proves unreliable due to substantial inconsistencies in the model' s self-generated descriptions. Our final prompt (Figure C.5) instead requires the model to jointly predict the current assembly step, the two toy parts being assembled, and the remaining unassembled components, ignoring irrelevant elements. The prompt additionally includes explicit warnings designed to reduce confusion among visually similar parts.

```
Analyze pictures of assembling toys. Toy parts are from the list: {
    combined_object_text}.
List of possible steps: {steps}.
Ignore small bolts or pegs, ignore hands/forearms working on the toy.
DO NOT CONFUSE BODY AND CHASSIS AND CABIN AND BASE.
Recognize the assembling parts, step, and unassembled toy parts on the
    table.
Write only json file:
{
    ''step_name'':''<step_name>'',
    ''assembling_objects'':''list 2 assembling objects'',
    ''unassembled_toy_parts'':''list unassembled toy parts you see on the
        table''
}
```

Figure C.5: Assembly101 prompt

Table C.10: Comparison of multimodal understanding across three tasks: **Action Detection**, **Relevant Object Detection**, and **Step Description Prediction from Video** on **CaptainCook4D**.

Model	Action Detection		Relevant Object Detection (3 Frames)					Step Description Prediction	
	Exact	Sim (0.98)	1-Match	2-Match	3-Match	Exact	Precision (Mean± Std)	Exact	Sim (0.8)
Gemma 3	0.323	0.323	0.891	0.658	0.409	0.015	0.235 ± 0.203	0.315	0.315
GPT-4o	0.601	0.601	0.853	0.570	0.320	0.069	0.388 ± 0.294	0.597	0.597
Gemini	0.710	0.738	–	–	–	–	–	0.655	0.655
InternVL3	0.208	0.208	–	–	–	–	–	0.328	0.328

C.5.3 VLM Model And Experimental Details

Gemma-3 (google/gemma-3-4b-it) was used through the HuggingFace image-text-to-text pipeline on GPU with bfloat16 precision. InternVL3 (OpenGVLab/InternVL3-8B-hf) was loaded with 4-bit quantization when available, otherwise in fp16. InternVL3 processed 16 uniformly sampled frames per video segment. Gemini inference used gemini-2.5-flash with default API settings. GPT-4o and GPT-5 were queried through the OpenAI API with default parameters.

We constructed curated evaluation splits for each dataset to assess VLM performance and select the best model for each setting. For Assembly101, it consists of 15 randomly selected videos featuring erroneous assembly attempts across different toys (202 steps total). For CaptainCook4D, it includes 48 randomly selected videos, balanced between 24 erroneous and 24 correct ones. Although we explored multiple frame-selection strategies, all reported results use three frames per video segment. For CaptainCook4D, we use the three central frames extracted from five uniformly sampled frames, which provides the most stable predictions across tasks. For Assembly101, three uniformly sampled frames per segment provide the

Table C.11: Comparison of **Step Prediction**, **Assembling Object Prediction**, and **Complementary Object Prediction** on **Assembly101**.

Model	Step Prediction		Assembling Object Prediction (2-Objects)			Complementary Object Prediction	
	Exact Match	Sim (0.98)	Exact Match	Similarity Match	1-Match	Exact Match	Similarity Match
GPT-5 (3 frames)	0.333	0.348	0.377	0.396	0.744	0.082	0.222
Gemini (segments)	0.227	0.251	0.285	0.295	0.710	0.048	0.155

Mod.	Encoder	Dim	Emis.	Align.	SeqAcc	Any	T-Aware	Video	Ins.	Del.	Subst.	Trans.
T	all-mpnet	256	Gauss	nw	0.741	0.766	0.696	0.990	0.890	0.996	0.888	0.989
T	all-minilm	384	Gauss	difflib	0.741	0.765	0.694	0.984	0.889	0.996	0.886	0.989
T	openai	768	Gauss	difflib	0.741	0.989	0.691	0.990	0.886	0.996	0.894	0.989
V	videomae-v2	64	Cosine	difflib	0.158	0.515	0.577	1.000	0.791	0.899	0.530	0.981
V	internvl3	1024	Gauss	difflib	0.157	0.506	0.640	1.000	0.571	0.891	0.957	0.980
V	dinov3	64	Gauss	difflib	0.136	0.500	0.573	1.000	0.791	0.898	0.526	0.974

Table C.12: Legacy Assembly101 PCMD results from the earlier ChronoFix study. These numbers are reported under the original accuracy-based evaluation used in that study and are therefore kept separate from the unified Any-F1 / Type-F1 results in the main text.

most reliable performance.

Table C.10 reports results on CaptainCook4D. We evaluate **Step Description Prediction** and **Action Recognition** using Exact Match (Exact) and Similarity Match (Sim) metrics. Similarity is computed based on the InstructXL results by the following instructions to embed steps and actions:

1. "Represent the cooking action (verb + object), focus on what action is done and what objects are used; do not ignore quantities and timing."

2. "Represent the cooking action, focus on what action is done (verb)."

These embeddings capture fine-grained nuances in both actions and step descriptions. For Similarity Match, a prediction is counted as correct if its similarity exceeds thresholds of 0.98 for step descriptions and 0.80 for actions. Thresholds were chosen to allow minor hallucinations while remaining higher than the maximum similarity across all other steps and actions. Relevant Object Detection is measured by the number of correctly predicted relevant objects, reported as at least 1-Match, 2-Match, 3-Match, and Exact Match along with Object Precision (Mean $\pm$ Std). For each step i, we compute

$$\text{Precision}_i = \begin{cases} \frac{\text{matched\_count}_i}{|\text{Pred}_i|}, & \text{if } |\text{Pred}_i| > 0, \\ 0, & \text{otherwise}, \end{cases} \tag{C.1}$$

where Pred_i is the set of predicted relevant objects. We report the *mean* and *standard deviation* of Precision_i across all steps.

Table C.11 shows results on Assembly101. For Step Prediction, we use the same metrics as for CaptainCook4D: Exact Match and Similarity (Sim) Match, but with the instruction for InstructXL: "Represent the assembly instruction step, focus on objects (toy parts)" and the same threshold of 0.98. For Assembling Object Prediction (2-Objects), we report Exact Match and Similarity Match (threshold 0.98), as well as 1-Match, which counts a prediction as successful if at least one object is correct. Exact Match is particularly important here, as each step involves exactly two objects, unlike CaptainCook4D. For Complementary Object Detection, we use Exact Match and Similarity Match (threshold 0.98) to assess how accurately the model identifies all unassembled toy parts present on the table. Apart from Gemini and GPT-5, we also experimented with open-source VLMs such as Gemma-3; however, performance was very low (Exact Match: 0.0773 for Step Prediction, 0.0918 for Assembling Object Prediction), so these results are not included in the table.

C.6 Finetuning Experiments

We attempted several finetuning strategies on the CaptainCook4D and Assembly101 training splits (about 115-117 videos) to make a model distinguish fine-grained procedural steps via multiclass video classification, by aligning each visual segment with a specific step description. None of these attempts gave meaningful generalization because each step label has very few examples and the label distribution is strongly long-tailed.

First, we finetuned visual encoders for step prediction. We tried CLIP and DINOv3 (three sampled frames), VideoMAE, and VideoMAE2, and also added LoRA adapters to the query/-value layers. Validation accuracy stayed extremely low (val acc $\approx$ 3–4%), with heavy class imbalance. The models did not see enough diverse examples per step within each chunk, and the full models overfit quickly.

We then tested alternative training objectives. We applied triplet loss (anchor-positive-negative) [108] between steps to reinforce step separation, but the data are too limited and many steps are visually close, so the model still memorized the training set. We also trained a VideoMAE2 encoder with an MLP classifier. It reached near-perfect training accuracy, but validation performance dropped fast even with strong regularization, again showing overfitting. Using only visual features did not help: an MLP that tries to separate about 350 step classes is too weak under such imbalance, and the embeddings are not discriminative enough for this taxonomy.

To reduce discreteness, we trained an MLP on CLIP visual embeddings to regress to CLIP text embeddings of step descriptions. This reproduced training embeddings well but failed on held-out segments, especially for rare steps. We also tried QLoRA finetuning of InternVL3-8B, but the model quickly memorized the training data and did not generalize. Finally, we trained an XGBoost classifier on VideoMAE2 embeddings for action and object prediction. This avoided strong overfitting, but test accuracy was still very low ($\approx 15\%$), which again shows that the visual embeddings alone are not sufficient.[1]

Overall, these results show that without text support it is hard to learn step semantics from so few videos, with many visually similar steps and, in CaptainCook4D, large visual variation of the same step. This is why we eventually rely on prompt-based inference with pretrained VLMs instead of task-specific finetuning.

[1]For Assembly101, we also trained an XGBoost classifier on VideoMAE2 embeddings to predict assembled objects but it shows the same behavior on test data. Some toy parts appear only in error videos or in a single video, so reliable learning is complicated.