Politecnico di Torino

Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Informatica (Computer Science)

A.a. 2025/2026

Graduation Session March 2026

CLIP-MD

Causal and Low Latency Inference for Procedural Mistake Detection

Relatori:

Francesca Pistilli

Giusepp Averta

Gaetano Falco

Candidati:

Jacopo Spaccatrosi

Abstract

Procedural Mistake Detection (PMD) aims to identify deviations from an expected multistep workflow while a user is performing a task. CLIP-MD (Causal and Low Latency Inference for Procedural Mistake Detection) is the framework proposed in this thesis to address PMD in egocentric settings, where systems must operate with strictly causal inference, low latency, and robustness to visual noise such as occlusions and fine-grained hand–object interactions. In this work, PMD is formulated as a One-Class Classification problem: models are trained only on correct executions and must detect deviations at inference time.

CLIP-MD adopts a dual-branch online pipeline. Firstly, a step recognition branch performs Online Action Detection (OAD), producing a causal stream of recognised procedural steps from video observations. Secondly, a step anticipation branch predicts the next expected step from the recognised history; a mistake is detected when executed and expected steps disagree.

For online action recognition, we analyse compact trained models and agnostic Multimodal Large Language Model (MLLM) approaches, showing that the latter remain unsuitable for OAD due to limited fine-grained accuracy and low throughput. We therefore introduce two causal OAD architectures. The Online Graph-Temporal Network (OGTN) combines latent object-centric slots, compact intra-frame relational reasoning, and efficient temporal modelling on compressed representations. The other proposed model is the Online Temporal Transformer (OTT), which uses windowed causal self-attention with multi-scale temporal pooling to improve temporal stability while preserving efficiency. Across both datasets, OGTN provides the best balance between accuracy, robustness, and throughput.

For step anticipation, we explore both agnostic LLM prompting strategies and lightweight Transformer predictors trained on procedural sequences. Prompting encourages procedural reasoning while maintaining a single step output suitable for online deployment, while trained models learn transition priors that capture procedural dynamics from symbolic histories under causal constraints.

Finally, we study frame-to-step aggregation, a key latency bottleneck in online procedural analysis, evaluating causal strategies based both on predicted labels and on model logits to reduce latency while preserving temporal stability.

Overall, by analysing and improving all modules of the PMD pipeline, CLIP-MD achieves a significantly more efficient online mistake detection process than the baseline, maintaining comparable or improved results with faster response times and more compact architectures

Acknowledgements

I would like to express my sincere gratitude to all the people who supported me throughout this journey.

First and foremost, I would like to thank my girlfriend, my family, and my friends for their constant presence and unwavering support. Their patience, understanding, and encouragement helped me face the challenges of this path with greater strength and confidence. In the most difficult moments, their belief in me reminded me why I started and gave me the motivation to keep moving forward. Without them, reaching this milestone would have been far more difficult.

Finally, I would like to thank myself for having the perseverance to continue even when the challenges felt overwhelming and the temptation to give up was strong. Completing this work is not only the conclusion of an academic journey, but also a personal reminder of the importance of determination and resilience.
This thesis stands as a small proof to myself that consistency and commitment can overcome doubts and difficulties. I hope that, in the future, I will be able to look back at these pages and remember that when I truly dedicate myself to something, I have the strength to carry it through to the end.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Procedural activities are part of everyday life. From cooking a simple meal to assembling furniture or repairing a bicycle, people constantly follow sequences of actions that must be executed in the correct order. In professional environments, this requirement becomes even more critical. In manufacturing, medical procedures, or mechanical assembly, performing steps incorrectly or out of sequence can compromise safety, reduce quality, and increase operational costs.

The availability of wearable cameras and augmented reality devices has made it possible to observe these activities directly from the performer's perspective. Egocentric video offers detailed visual information about hand movements, object manipulations, and tool usage. This opens the possibility of building systems that monitor ongoing procedures and provide timely feedback when something goes wrong, for example when a step is skipped, repeated, or executed prematurely. Such systems could support training, reduce errors, and improve reliability in high-risk contexts.

Over the past years, the computer vision community has made significant progress in understanding procedural video. Several datasets have been introduced to study action segmentation, task understanding, and mistake detection, including large-scale instructional collections and egocentric assembly recordings. However, approaches to **procedural mistake detection (PMD)** remain heterogeneous. Some methods rely on explicit action recognition and compare predicted steps against a predefined workflow, while others focus on object state changes to infer inconsistencies. This variety reflects the complexity of the problem, but it also makes systematic comparison difficult. There is still a need for unified frameworks that allow mistake detection methods to be evaluated under consistent assumptions and realistic online constraints.

1.1 PMD and its Requirements

An effective **Procedural Mistake Detection (PMD)** system must operate reliably across diverse error scenarios. In practical settings, mistakes rarely follow a single pattern: a step may be executed out of order, skipped entirely, repeated unnecessarily, or replaced by an action that does not belong to the intended workflow. Limiting detection to a fixed list of predefined error types would therefore be unrealistic. Instead, the system should be able to recognise any deviation from the expected procedural structure.

For this reason, this thesis adopts a **One-Class Classification (OCC)** formulation for procedural mistake detection. The model is trained exclusively on correct executions and learns to capture the structure of valid workflows without requiring explicit labels for individual mistake categories. At inference time, deviations from the learned procedural dynamics are treated as potential errors. This choice reflects a practical constraint: collecting exhaustive annotations for all possible mistake types is expensive and often infeasible, as errors are rare and highly variable by nature. Framing PMD as an OCC problem allows the system to focus on modelling what is correct, rather than enumerating what can go wrong, and promotes consistent evaluation across different procedural domains.

In addition to robustness, timeliness is essential. A PMD system should detect deviations as soon as they occur, ideally while the action is still being performed, so that corrective feedback can be provided immediately. Such systems can operate in either online or real-time settings. In both cases, predictions must remain causal: only current and past observations can be used, without access to future frames. Online systems process data sequentially and tolerate small, bounded delays (e.g., $\geq$ 50ms) that do not affect usability. Real-time systems, instead, impose strict end-to-end latency guarantees (e.g., $\leq$ 50ms), with hard constraints dictated by the sensing–processing–feedback loop. For most wearable use cases, enforcing strict real-time guarantees would introduce unnecessary constraints, since small delays are generally acceptable and do not compromise the feedback loop.

1.2 Dual-Branch Architectures for PMD

Recent state-of-the-art approaches adopt a dual branch architecture for online procedural mistake detection. These architectures consist of:

- **Step Recognition Branch:** A visual recognition module performs online action detection, producing frame level predictions of the ongoing video.
- **Step Anticipation Branch:** A reasoning module predicts which step should come next, following the sequence of recognized steps.

As illustrated in Figure 1.1, at each frame t both branches produce a step prediction. The recognition branch estimates the action currently being executed, while the anticipation module predicts the step that should occur next according to the learned procedural dynamics. A mistake is detected whenever the observed action becomes inconsistent with the expected one.

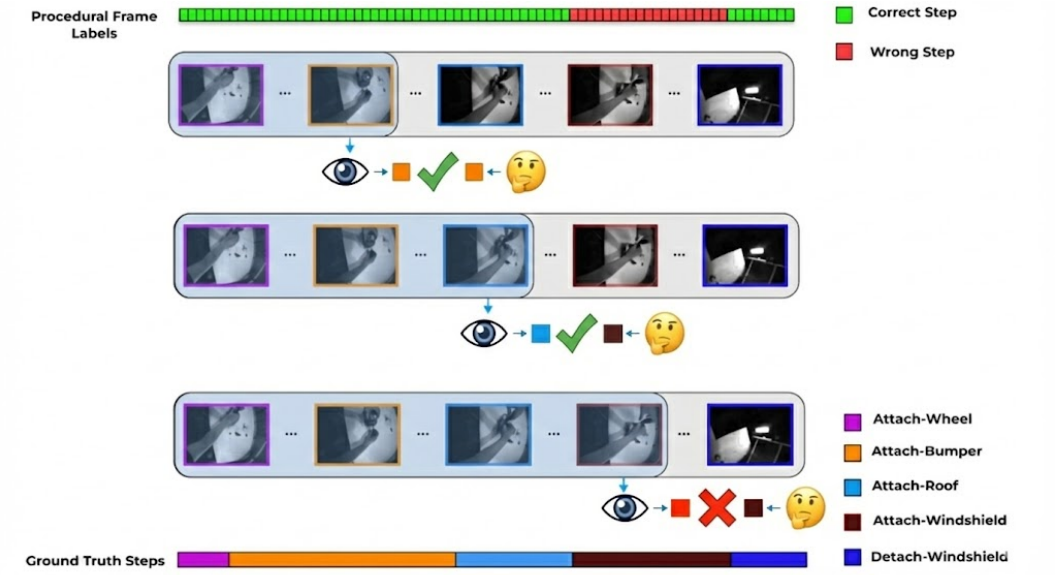


Figure 1.1: Dual-branch online PMD. For each frame, the recognition branch predicts the executed step, while the anticipation branch estimates the expected one. A mismatch between the two triggers a mistake signal

This formulation offers two main advantages. Firstly, it naturally aligns with the OCC setting, as training requires only correct procedural sequences without explicit modelling of erroneous cases. Secondly, it clearly separates perceptual recognition from procedural reasoning: the recognition branch focuses on visual understanding of the current action, while the anticipation branch models the temporal structure of the procedure, enabling a clean separation of responsibilities within the pipeline.

1.3 Limitations of Existing PMD

When examined under realistic online constraints, existing Procedural Mistake Detection (PMD) pipelines expose a number of structural limitations.

Recognition Stability under Egocentric Dynamics
In online PMD systems, recognition models are typically kept lightweight to ensure fast inference. While this enables efficient frame processing, it often limits temporal modeling capacity. In egocentric video, characterized by motion blur, occlusions, and rapid viewpoint changes, this constraint can lead to unstable predictions that propagate to the segment level, producing inconsistent boundaries and fragmented action sequences that undermine mistake detection reliability.

Aggregation Latency As A Bottleneck in Online PMD
In many online PMD pipelines, aggregation becomes a major source of delay. Step predictions are typically confirmed only after collecting multiple frame level outputs, which improves stability but postpones the final decision. As a result, feedback about an incorrect action may arrive several seconds after the action has started, reducing system responsiveness and limiting usability in interactive settings.

Prompting Strategies Not Fully Exploiting LLM Reasoning
While chain-of-thought prompting has been shown to improve structured reasoning in procedural anticipation, existing prompting strategies are often generic and insufficiently tailored to procedural transitions. In particular, they rarely exploit more advanced reasoning mechanisms such as hypothesis comparison, self-verification, or multi-candidate evaluation, which could increase robustness in ambiguous or visually uncertain situations.

Underexplored Trained Anticipation Models
Many existing PMD pipelines rely on a single anticipation strategy, typically based on large pre-trained models or predefined reasoning schemes, without investigating alternative trainable solutions in depth. As a result, task-specific anticipation architectures have received comparatively little attention, and direct comparisons across different model designs are rare. This makes it difficult to assess how architectural choices impact robustness, latency, and online suitability.

Inference and Deployment Cost
Recent PMD approaches increasingly rely on large-scale models, often comprising several billions of parameters, for step anticipation. While effective in terms of predictive accuracy, such models introduce substantial computational cost and inference latency. This reliance primarily affects the anticipation stage and limits the feasibility of real-time deployment, especially on smaller or resource constrained systems.

1.4 Contributions of Thesis

This thesis introduces improvements across all three major components: step recognition, frame-to-step aggregation, and anticipation module.

Novel OAD Models (Online Action Detection)
We propose two novel recognition models:

- **Online Graph-Temporal Network (OGTN):** A graph-based architecture that combines multi-head attention with latent-slot representations. OGTN integrates frame level evidence with causal temporal context, coupling appearance cues and structured temporal reasoning to produce stable predictions. The model is designed for efficiency, operating on compact representations and introducing temporal modeling only when needed. Despite its lightweight structure, OGTN matches or surpasses state-of-the-art performance on both datasets while achieving higher throughput.

- **Online Temporal Transformer (OTT):** An object-centric, fully attention-based recognition model in which each frame is encoded as a compact token and processed by a causal temporal Transformer. Temporal reasoning is performed through windowed attention over past frames, ensuring strict online operation, and is complemented by short- and mid-term temporal pooling to enhance stability. Despite relying exclusively on attention mechanisms, OTT remains computationally efficient and achieves performance comparable to the baseline across both evaluated datasets.

Advanced Frame-to-Step Aggregation
Multiple frame-to-step aggregation methods were evaluated to obtain step sequences closely aligned with the ground truth while minimising commit latency in a strictly online setting.

- **Label-based strategies:** Deterministic aggregation methods operating directly on discrete frame predictions.
  - **Minimum-Duration Filtering + NOMA (MDF+NOMA):** applies a minimum persistence constraint to validate label transitions before performing block-based majority voting (NOMA).
  - **Hysteresis decoding:** introduces temporal inertia by delaying state transitions until sufficient evidence for a new label has accumulated, suppressing short-lived oscillations.

- **Logit-based strategies:** Confidence-aware aggregation methods operating directly on frame logits, accumulating soft evidence over time to improve stability at reduced latency.
  - **Soft-OMA:** performs overlapping aggregation in logit space by averaging log-probabilities over a trailing causal window.
  - **Exponential Moving Average (EMA):** smooths class probabilities over time through incremental causal updates.
  - **Viterbi with Transition Penalty (VTP):** accumulates log-evidence sequentially while discouraging frequent label switches through an explicit transition cost.

Enhancing Procedural Step Anticipation
The step anticipation module is explored along two directions. On one side, agnostic LLM anticipation is extended with structured prompting strategies that introduce controlled reasoning while maintaining a symbol output suitable for online deployment. On the other, we investigate task trained predictors that learn procedural transition dynamics directly from annotated step sequences, with models:

- **Causal RoPE Anticipation Transformer (CRAT):** A trained Transformer model for fully causal next-step prediction over symbolic action sequences. Temporal order is encoded through Rotary Positional Embeddings (RoPE).
- **Causal RoPE Anticipation Transformer with Graph Bias (CRAT-GB):** An extension of CRAT with a graph transition prior learned during training, guiding predictions toward plausible step transitions.

Scalability and Deployment Improvements
Recent PMD approaches increasingly rely on large scale anticipation models and aggregation schemes with commit delays of several seconds, resulting in high computational cost and latency. This limits deployment on resource constrained systems and reduces suitability for online applications.
In this work, scalability and deployability are considered throughout the entire pipeline. Lightweight recognition models, latency-aware aggregation, and compact trained anticipators are jointly designed under strict causal constraints,
yielding a modular PMD framework that preserves competitive performance while substantially reducing computational overhead and latency.

Chapter 2

Related Works

Procedural mistake detection lies at the intersection of several research areas, including temporal action recognition, anomaly detection, structured graph learning, and sequence modeling with language models. Rather than constituting an isolated research direction, it builds upon methodologies originally developed for online perception and structured reasoning.
This chapter reviews prior work by focusing on the modeling assumptions adopted to represent procedural consistency. We begin by examining how correctness has been formalized in mistake detection frameworks (Sec. 2.1), then move to online step recognition models within the Online Action Detection (OAD) paradigm (Sec. 2.2), and conclude with anticipation and reasoning mechanisms for forecasting future procedural steps (Sec. 2.3).

2.1 Modeling Procedural Consistency

A central issue in procedural mistake detection concerns the formalization of procedural validity, namely how correct task progression is represented and how deviations from it are identified. Existing approaches differ in the modeling assumptions adopted to capture sequential dependencies and in the criteria used to determine inconsistency.

Closed-set error classification

A first line of work frames procedural mistake detection as a supervised classification problem under closed-set assumptions. Models are trained on annotated examples of both correct and incorrect executions and assign each observed sequence to a predefined error category. Benchmarks such as ATA[1], Assembly101[2], EPIC-Tent[3] and HoloAssist[4], follow this paradigm, providing manually annotated deviations from reference procedures to enable explicit modeling of mistake types in structured manipulation and assembly tasks.

In this formulation, correctness is defined with respect to a fixed error taxonomy, and the objective is purely discriminative, requiring all mistake categories to be specified during training. While effective for known errors, this closed-set setting limits generalization to unseen deviations. Several frameworks follow this paradigm; for instance, EgoPED[5] models procedural deviations using action segmentation and contrastive prototypes over correct and erroneous steps. Surveys such as Vision-Based Mistake Analysis in Procedural Activities[6] review vision approaches for detecting mistakes in procedural tasks, categorizing methods by supervision level, learning strategy, and use of procedural structure.

One-class and anomaly-based modeling

To improve generalization beyond predefined error types, recent work frames procedural mistake detection as a one-class problem, training models only on correct executions and identifying deviations at inference time. This view relates to deep one-class methods such as Deep SVDD[7], which learn compact representations of normal data, and to video anomaly detection[8, 9], where regular temporal patterns are modeled without explicit anomaly labels. Unlike generic anomaly detection, however, procedural mistakes arise from violations of task progression rather than purely visual irregularities, requiring reasoning over action relationships.
PREGO[10] adopts this perspective through an online OCC framework and adapted benchmarks (Assembly101-O and EpicTent-O) derived from Assembly101[2] and EPIC-Tent[3], enforcing causal and one-class protocols. TI-PREGO[11] further refines this setting by addressing class imbalance and strengthening temporal aggregation prior to higher level reasoning.

Graph-based structural modeling

A complementary paradigm models procedural consistency through explicit graph representations, building on prior work in spatio-temporal graph modeling for action recognition. For instance, ST-GCN [12] represents human motion as a graph of joints connected across space and time, showing how explicit connectivity can improve representation learning. Recent works such as Skeleton Graph Contrastive Learning [13] and SimCLR [14] extend this idea by applying contrastive objectives to learned graph embeddings, encouraging the model to separate different motion patterns while preserving the underlying structural relationships. Extending these concepts to procedural reasoning, Differentiable Task Graph Learning (TGML) [15] models task structure directly from data by learning a task graph as a parameterized adjacency matrix optimized end-to-end. Here, nodes correspond to procedural steps and edges encode valid transitions or preconditions, allowing mistake detection to be framed as identifying violations of learned structural dependencies.

2.2 Online Step Recognition

Identifying the currently executed procedural step is a core component of online procedural mistake detection. In streaming settings, this task is addressed within the Online Action Detection (OAD) framework, which recognizes actions causally as frames become available.

Recurrent Streaming Models

Recurrent Neural Networks (RNNs[16]), including LSTMs[17] and GRUs[18], have traditionally played a central role in Online Action Detection (OAD), mainly because their causal structure allows predictions to be produced frame by frame without relying on future information.
Early work, such as Joint Classification-Regression RNNs[19], uses LSTM[17] models to jointly classify actions and estimate temporal boundaries from streaming data. The Temporal Recurrent Network (TRN)[20] moves in a similar direction, combining online detection with short term anticipation in a unified framework. More recently, MiniROAD[21] shows that relatively simple GRU[18] architectures can remain competitive when training explicitly mirrors the streaming scenario, in which predictions are generated frame by frame using only limited past context.

Attention based Temporal Models

Transformer architectures[22] provide an alternative to recurrent models by capturing long range temporal dependencies through self-attention.
In Online Action Detection (OAD), attention models process past frames causally, allowing each prediction to draw on a weighted temporal context.
OadTR[23] adopts an encoder–decoder design in which self-attention encodes historical observations and a cross-attention decoder generates action representations for online prediction. LightTR[24] focuses on efficiency, restricting attention to historical and current frames and using action specific queries to refine outputs while reducing computational cost.
CMeRT[25] addresses the gap between training and streaming inference by enhancing contextual encoding of past frames and introducing a causal short term memory module, improving stability and consistency in online predictions.

Structured Priors in Representation Learning

Some works move beyond purely temporal modeling and try to encode structure directly in the representation. Instead of treating frames as isolated inputs, they model relations within the scene.
ST-GCN[12] represents human motion as a graph of joints connected across space and time, letting the network learn which connections matter for coherent motion patterns. Contrastive approaches such as Skeleton Graph Contrastive Learning[13], together with contrastive objectives[14], strengthen these graph features by pushing apart representations of different motions.
A similar intuition appears in object-centric models. Slot Attention[26] decomposes visual input into a fixed set of latent slots that tend to capture individual objects. Perceiver[27] processes high dimensional inputs through a compact latent bottleneck using cross-attention, making large inputs more manageable.

2.3 Anticipation and Sequential Reasoning

Procedural coherence depends not only on recognizing the current step, but also on predicting its progression. Action anticipation supports this process by detecting mismatches between observed and expected transitions.

Predictive Sequence Modeling

Predictive sequence modeling is often used to anticipate the next step in structured procedures, treating the problem as the continuation of a sequence based on past observations. In language modeling, this idea appears in different forms.
BERT[28] relies on masked language modeling to learn bidirectional contextual representations, capturing dependencies within a sequence.
Autoregressive transformers[22], instead, generate tokens sequentially through next-token prediction, which aligns more naturally with causal forecasting in online settings. RoFormer[29] builds on this framework by introducing Rotary Position Embeddings, improving the handling of relative positions and maintaining stability over longer contexts. This aspect is particularly relevant in procedural reasoning, where the validity of a step depends on its position within an ordered and temporally consistent sequence.

Structural Constraint Reasoning

An alternative to purely probabilistic sequence prediction models anticipation through explicit structural constraints. Early hierarchical task graph approaches, such as the work by Lu and Elhamifar[30], describe procedures as directed acyclic graphs where valid future steps are determined by enforcing prerequisite and ordering relations encoded in the graph. In their method, procedural structure is learned through set-supervised action learning with pairwise order consistency, allowing the model to infer valid step dependencies directly from instructional videos. This idea is further developed in spatio-temporal scene graph models such as Action Genome[31], which represent objects and their interactions over time to support relational reasoning. Egocentric Action Scene Graphs[32] adapt this representation to long egocentric videos, explicitly modeling actions, objects, and semantic relations to enable more structured long-term anticipation.
Graph-based anticipation frameworks like TGML[15] apply a similar principle at the procedural level: instead of directly predicting the next action, the model learns structural dependencies between steps and encodes admissible transitions in a task graph. Anticipation then amounts to verifying whether observed transitions comply with these learned constraints.

LLMs Symbolic Forecasting

Large Language Models (LLMs) bring predictive sequence modeling closer to symbolic reasoning. By learning structured token dependencies at scale, they can operate directly in discrete procedural spaces and support next-step forecasting.
PREGO[10] and TI-PREGO[11] follow this direction, using LLMs as symbolic predictors in the label space. Through in-context learning[33] and chain-of-thought prompting[34], the model conditions on the recognized sequence of steps and generates a plausible continuation. A procedural mistake is detected when the observed transition does not match the predicted one.
This formulation separates perceptual recognition from symbolic forecasting, keeping visual grounding distinct from task level reasoning. Such a modular design facilitates flexible modeling of procedural dynamics and naturally accommodates open-set scenarios, where unseen deviations cannot be exhaustively specified during training.

Chapter 3

Datasets

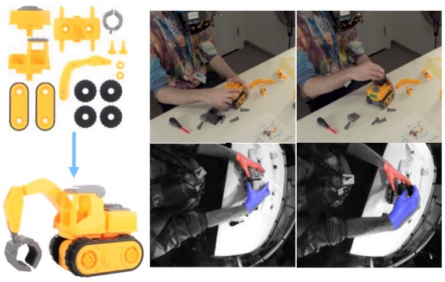
To ensure a fair comparison with previous work, this thesis adopts the same datasets used by the reference baseline for online procedural mistake detection.
Specifically, we rely on Assembly101-O and Epic-Tent-O, online variants of Assembly101[2] and Epic-Tent[3] introduced in PREGO[10] to support causal evaluation and open-set mistake detection.

Using the same benchmarks allows the proposed approach to be evaluated under identical conditions and facilitates a direct comparison with existing methods in the literature. In this work, we consider exclusively the **egocentric video**, recordings captured from the performer's point of view.
This perspective is particularly suitable for procedural understanding, as it emphasizes hands, tools, and object interactions, closely reflecting the visual information available during task execution. Consequently, egocentric footage provides strong cues for identifying step boundaries, action intent, and object manipulations, making it well aligned with online recognition and mistake detection.

The online variants of Assembly101[2] and Epic-Tent[3] follow a **One-Class Classification (OCC) paradigm**. The training split contains only correct procedural executions, enabling the model to learn the canonical workflow without exposure to errors. In contrast, validation and test splits include sequences with procedural mistakes, which must be detected as deviations from the learned normal behaviour. This setup supports a genuine open-set evaluation and reflects realistic deployment scenarios, where it is impractical to enumerate all possible procedural deviations. To comply with online constraints, test videos are evaluated only up to the first mistake, ensuring that downstream modules operate on procedurally valid context. Together, Assembly101-O and Epic-Tent-O enable the evaluation of the three core components of the proposed pipeline: online action recognition, causal step anticipation, and the detection of previously unseen procedural errors.

3.1 Assembly101-O

Assembly101[2] is one of the largest and most comprehensive datasets for procedural video understanding. It contains hundreds of mechanical assembly and disassembly tasks performed on miniature vehicles, recorded from multiple viewpoints and annotated at several levels of granularity. In this work, only the egocentric camera views are considered, since the proposed PMD pipeline operates under first person visual assumptions.



Assembly101-O is a variant explicitly designed for online and open-set mistake detection following the One-Class Classification (OCC) paradigm. In this reinterpretation, the dataset is reorganized so that the training split contains only correct procedures. The model is therefore exposed exclusively to valid workflows, learning the canonical sequence of steps that characterize a successful assembly. Conversely, the validation splits include only sequences containing mistakes. Since these errors are never seen during training, the model must detect them as deviations from the learned normal behaviour, enabling a true open-set evaluation.

Assembly101-O is particularly well suited for this thesis due to its intrinsic procedural complexity. The dataset comprises 86 distinct actions arranged in structured assembly workflows, making the modelling of correct step transitions and procedural consistency highly challenging. Its per-frame annotations enable precise identification of deviations from the canonical workflow, which is essential for procedural mistake detection. The combination of an egocentric viewpoint, strict OCC training conditions, and a large and diverse action space provides all the necessary components for evaluating procedural mistake detection.

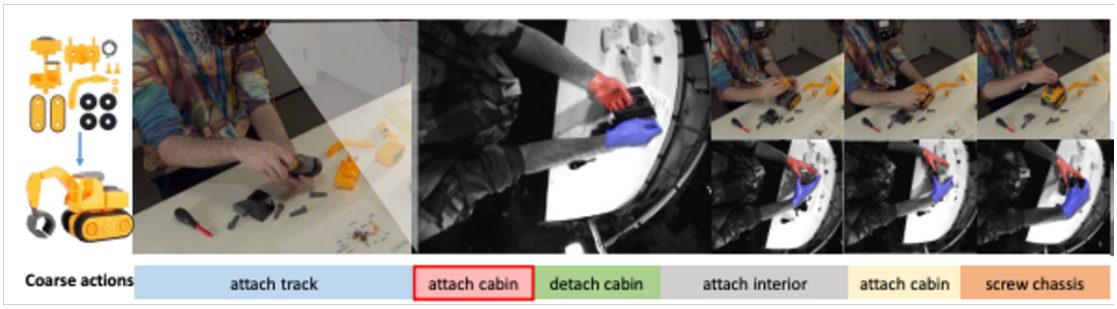


Figure 3.1: Example from Assembly101 showing fixed and egocentric views, with the corresponding action annotations.

3.2 Epic-Tent-O

Epic-Tent[3] is an egocentric dataset that captures the assembly of a camping tent in outdoor environments. All recordings are acquired using head-mounted GoPro cameras, providing a first-person view that closely reflects real-world procedural scenarios.
The dataset includes per-frame step annotations, nine categories of mistakes, and self-reported uncertainty scores provided by the performers during task execution.

A key difference between Epic-Tent[3] and Assembly101[2] lies in the distribution of mistakes. While Assembly101[2] includes several fully correct procedures, every Epic-Tent[3] video contains at least one mistake, preventing its direct use in a One-Class Classification (OCC) setting. To address this limitation, Epic-Tent-O uses the **self-reported uncertainty scores** provided by performers as a proxy for procedural correctness. These scores are part of the original dataset annotations, allowing videos with higher confidence to be used for training, while those with lower confidence are reserved for validation and testing.

Although Epic-Tent defines nine mistake categories, only those that invalidate the procedural workflow are considered in Epic-Tent-O. Evaluation is restricted to order, omit, correction, and repeat, while categories such as slow, motor, search, and misuse are excluded as they do not disrupt procedural consistency.

Epic-Tent-O is well suited for evaluating anticipation modules in procedural mistake detection. The dataset includes 12 structured high-level steps, while naturally occurring mistakes and the outdoor egocentric setting introduce realistic variability, making it a challenging benchmark for open-set detection.

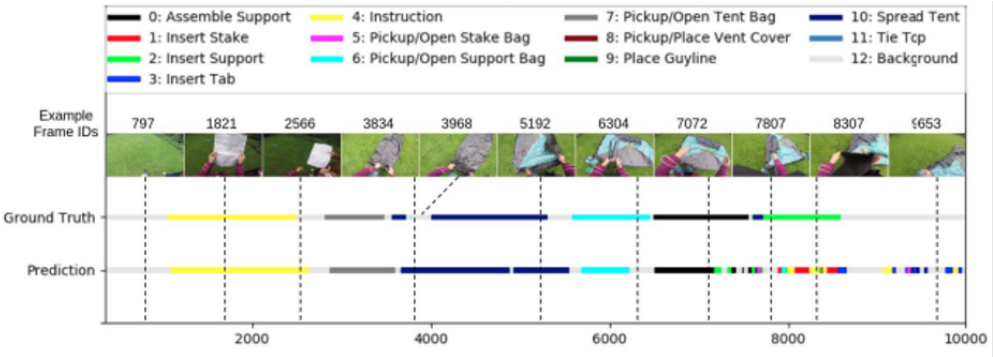


Figure 3.2: EpicTent example with fixed and egocentric views and step annotations

Chapter 4

Task Definition

Online procedural mistake detection is formulated as a pipeline composed of three interconnected tasks: step recognition, temporal aggregation, and step anticipation. The interaction between these components enables the detection of procedural deviations in a causal and online setting.

4.1 Online Action Detection (Step Recognition)

Given a video stream represented as a sequence of frames:

$$\mathbf{x}_{1:T} = (\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T) \tag{4.1}$$

the goal of **Online Action Detection (OAD)** is to predict, for every timestamp t, the step label:

$$\hat{y}_t^{\text{oad}} \in \mathcal{C} \tag{4.2}$$

where $\mathcal{C}$ is the set of procedural steps (86 for Assembly101-O, 12 for Epic-Tent-O).

The model must operate causally, meaning that each prediction is computed using only information available up to the current time step. Future frames or look-ahead context cannot be used, ensuring compatibility with online constraint.

$$\hat{y}_t^{\text{oad}} = f(\mathbf{x}_{1:t}) \qquad \hat{y}_t^{\text{oad}} \text{ may not depend on } \mathbf{x}_{t+1:T} \tag{4.3}$$

OAD model outputs are inherently noisy due to occlusions, motion blur, fine-grained hand–object interactions, and visual similarity between adjacent steps. Even within temporally coherent segments, the recogniser may produce spurious transitions, generating brief label fluctuations or premature step changes.
Such artifacts disrupt temporal consistency and negatively impact both anticipation and mistake detection. Therefore, a dedicated aggregation stage is required to stabilise frame-level predictions and recover a coherent step sequence.

4.2 Frame-to-Step Aggregation

The recognition module produces frame-level predictions $\hat{y}^{\text{oad}}_{1:T}$, which in egocentric video are often affected by noise, short-term ambiguities, and temporal instability. Rapid viewpoint changes, occlusions, and subtle action transitions frequently lead to label flickering, making the frame output prediction unsuitable for high-level procedural reasoning. To address this issue, a dedicated aggregation module converts the sequence of frame predictions into a compact and temporally consistent sequence of step segments:

$$\hat{s}_{1:K} = g(\hat{y}^{\text{oad}}_{1:T}), \qquad K \ll T \tag{4.4}$$

where K is significantly smaller than T, since consecutive frames assigned to the same procedural step are merged into a single segment. This transformation represents a crucial abstraction step, shifting the representation from frame predictions to a symbolic, step-based description of the ongoing procedure.
The output of this stage is an ordered sequence of stable step segments $\hat{s}_{1:K}$, which provides a symbolic representation of the observed procedure and serves as input to the step anticipation module.

4.3 Step Anticipation

Step anticipation addresses the problem of predicting the next expected action in an ongoing procedure, using only the sequence of steps recognised up to the current time. At time t, the system maintains an aggregated and temporally stabilised history of recognised steps $\hat{s}_{1:m(t)}$, where $m(t)$ denotes the number of step segments that have been produced up to that point. Starting from this symbolic history, the anticipation module estimates the most plausible next step:

$$\hat{y}^{\text{ant}}_t = A\Big(\hat{s}_{1:m(t)}\Big) \tag{4.5}$$

The anticipation step can be viewed as a procedural reasoning task: the system infers how a procedure typically unfolds and predicts the next action, even under partial observation. Unlike frame level recognition, which describes the present, anticipation reasons about structural ordering and long-term dependencies. This capability is crucial for mistake detection, since many errors emerge from deviations from the expected procedural flow.
Importantly, the anticipation module operates solely on symbolic information, without accessing raw visual data. By relying only on the recognised step sequence, perceptual uncertainty is decoupled from procedural reasoning, allowing the model to focus on high-level temporal logic.

4.4 Mistake Detection

A procedural mistake is detected whenever the step currently recognised by the system is inconsistent with the step anticipated as the next valid action. Formally, at time t, a mistake indicator is defined as:

$$\text{Mistake}(t) = \begin{cases} 1 & \text{if } \hat{y}_t^{\text{oad}} \neq \hat{y}_t^{\text{ant}} \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

where $\hat{y}_t^{\text{oad}}$ denotes the step recognised at time t, and $\hat{y}_t^{\text{ant}}$ represents the step predicted by the anticipation module as the expected continuation of the procedure. This decision rule captures the core intuition of procedural mistake detection, an error occurs when the observed execution deviates from the expected procedural flow.

Importantly, the system does not rely on explicit error annotations or predefined mistake categories. From a learning perspective, this formulation can be interpreted as a form of one-class learning. During training, the anticipation model is exposed only to correct procedural executions, allowing it to implicitly learn the structure and valid transitions of the procedure. At inference time, any deviation from these learned regularities is treated as a mistake.

Crucially, the system does not attempt to classify the type or cause of the error. Instead, it performs binary inconsistency detection, signalling when the ongoing execution becomes incompatible with the expected next step. This design enables the detection of both frequent and rare procedural mistakes, including those that were not explicitly observed during training.

Chapter 5

Evaluation Metrics

This chapter presents the evaluation metrics used to assess the proposed pipeline at three levels: recognition, aggregation, and anticipation.
Recognition metrics evaluate frame-level action predictions, aggregation metrics assess the quality of the resulting step sequence, and anticipation metrics measure the ability to predict the next expected step and detect deviations.

5.1 Recognition Metrics

To evaluate Online Action Detection (OAD), we assess how accurately the recogniser assigns an action label to each incoming frame under causal and online constraints. The evaluation is performed at frame level: given the ground-truth label y_t and the predicted label $\hat{y}_t$ at frame t, class-wise metrics are computed over the set of non-background classes $\mathcal{C}$. True positives (TP_c), false positives (FP_c), and false negatives (FN_c) are derived from frame-level assignments.

- **Accuracy:** Proportion of frames for which the predicted label matches the ground truth
$$\mathrm{Acc} = \frac{1}{T}\sum_{t=1}^{T}\mathbb{I}\left[\hat{y}_t = y_t\right] \tag{5.1}$$

- **Precision:** For each class, measures the fraction of frames predicted as that class that are correct; the reported value is averaged across classes
$$\mathrm{Precision} = \mathrm{mean}_{c\in\mathcal{C}}\left[\frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FP}_c}\right] \tag{5.2}$$

- **Recall:** For each class, the proportion of ground-truth frames correctly predicted as belonging to that class, averaged over all classes.
$$\mathrm{Recall} = \mathrm{mean}_{c\in\mathcal{C}}\left[\frac{\mathrm{TP}_c}{\mathrm{TP}_c + \mathrm{FN}_c}\right] \tag{5.3}$$

- **F1-score:** For each class, combines precision and recall through their harmonic mean; the reported value is averaged across classes

$$\mathrm{F1} = \mathrm{mean}_{c \in \mathcal{C}} \left[\frac{2\,\mathrm{Precision}_c\,\mathrm{Recall}_c}{\mathrm{Precision}_c + \mathrm{Recall}_c} \right] \tag{5.4}$$

- **Mean Average Precision (mAP):** Computed from the per-class confidence scores by summarising performance across all decision thresholds. For each class c, $AP_(c)$ is the average precision from its precision–recall curve; mAP is the mean over classes with at least one positive frame

$$\mathrm{mAP} = \mathrm{mean}_{c \in \mathcal{C}^+}(\mathrm{AP}_c) \tag{5.5}$$

- **Mean Class-AP (mcAP):** This corresponds to the mean of the calibrated average precision values (cAP_c), computed per class and then averaged over classes with at least one positive frame

$$\mathrm{mcAP} = \mathrm{mean}_{c \in \mathcal{C}^+}(c\mathrm{AP}_c) \tag{5.6}$$

5.2 Aggregation Metrics

The frame-to-step aggregation module defines the symbolic sequence used by anticipation and mistake detection. Its evaluation must assess how well the aggregated sequence matches the ground-truth procedure at the structural level.
Since aggregation is sequence based, frame metrics are inadequate; instead, errors in step ordering, presence, and segmentation must be considered.
To evaluate aggregation quality, we adopt the **Levenshtein Similarity**, a normalized metric derived from the Levenshtein distance for comparing symbolic sequences of variable length. Given two step sequences A and B, corresponding to the aggregated prediction and the ground truth, the similarity is defined as:

$$\mathrm{LevSim}(A, B) = 1 - \frac{d_{\mathrm{Lev}}(A, B)}{\max(\|A\|, \|B\|)} \tag{5.7}$$

where $d_{\mathrm{Lev}}(A, B)$ is the Levenshtein distance, i.e., the minimum number of insertions, deletions, or substitutions required to transform A into B. The score lies in [0,1], with higher values indicating better structural agreement.
Levenshtein Similarity measures aggregation quality by comparing the predicted and ground-truth step sequences at a symbolic level. It captures typical aggregation errors (e.g., missing or spurious steps) through the minimum number of edits required for alignment, while remaining largely insensitive to minor temporal boundary shifts. This makes it well suited for evaluating frame-to-step aggregation independently of downstream anticipation and mistake detection.

5.3 Anticipation Metrics

Step anticipation is formulated as a binary decision problem in which each time step is classified either as a correct continuation of the procedure or as a deviation. Positive samples correspond to deviations from the expected procedural progression (mistakes), while negative samples correspond to correctly executed steps that match the anticipated continuation.
Because procedural data is typically imbalanced, with mistakes occurring rarely, standard Precision metric can provide a distorted view of performance.
Following the TI-PREGO[11] strategy, we therefore report Balanced Precision and Balanced F1-score, which account for class imbalance by rescaling the contribution of false positives and offer a more reliable assessment of anticipation quality.

Let $N_+ = TP + FN$ denote the number of positive samples (mistakes) and $N_- = TN + FP$ the number of negative samples (correct steps). The balancing factor is defined as:

$$\alpha = \frac{N^+}{N^-} = \frac{\mathrm{TP} + \mathrm{FN}}{\mathrm{TN} + \mathrm{FP}}. \tag{5.8}$$

- **Balanced Precision:**

$$\mathrm{Precision}_b = \frac{\mathrm{TP}}{\mathrm{TP} + \alpha\,\mathrm{FP}}. \tag{5.9}$$

- **Balanced F1:**

$$\mathrm{F1}_b = \frac{2\,\mathrm{Precision}_b\,\mathrm{Recall}}{\mathrm{Precision}_b + \mathrm{Recall}}. \tag{5.10}$$

By explicitly accounting for the predominance of correctly executed steps, these balanced metrics prevent performance estimates from being dominated by the large number of negative samples, while still penalizing incorrect mistake detections.
As a result, they provide a fairer and more meaningful evaluation of step anticipation in realistic procedural scenarios.

Chapter 6

Recognition Models (OAD)

Online Action Detection (OAD) is a core component of procedural mistake detection, as it transforms a continuous video stream into a structured sequence of action labels on which aggregation and anticipation operate. Since these downstream stages rely directly on recognition outputs, inaccuracies at this level propagate forward, affecting temporal stability, symbolic coherence, and ultimately the reliability of deviation detection. For this reason, an effective recogniser must balance three aspects: predictive accuracy, temporal robustness (i.e., limited label flickering), and low latency inference suitable for online deployment.

This chapter presents the recognition strategies analysed in the thesis.
We first examine task-agnostic solutions, such as prompting Multimodal LLMs (e.g., Gemini[35], Qwen2.5[36], and InternVL2.5[37]) without task-specific training. In addition, we briefly explore a segmentation and tracking approach based on Meta's SAM 2.1[38], whose details are discussed separately in the ablation study (Section 9.1). While appealing for their generality, these approaches proved unsuitable for our scenario, yielding slower inference (lower FPS) and weaker action prediction accuracy on Assembly101-O and Epic-Tent-O.

We then describe the baseline adopted in prior work and introduce two novel architectures designed to better capture object-centric dynamics while remaining efficient under online constraints. All trained recognition models operate on pre-extracted visual features: RGB representations are obtained from a ResNet-50[39] backbone, while optical flow features are extracted using a BN-Inception[40] network pretrained on motion data. At each timestamp t, the input to the recogniser consists of the concatenation of RGB and flow embeddings extracted at that frame:

$$\mathbf{x}_t = [\mathbf{x}_t^{rgb}; \mathbf{x}_t^{flow}] \tag{6.1}$$

6.1 MiniROAD[21] (Baseline)

MiniROAD[21] is a lightweight GRU based recogniser designed to produce frame action logits with minimal computational overhead. The concatenated RGB and optical-flow features are first projected into a shared embedding space through a shallow MLP:

$$\mathbf{e}_t = \text{Dropout}(\text{ReLU}(\text{LayerNorm}(W\mathbf{x}_t + b))) \tag{6.2}$$

Temporal modelling is performed by a multi-layer GRU, which processes the embedded sequence in a strictly causal manner by recursively updating its hidden state over time:

$$\mathbf{h}_{1:t} = \text{GRU}(\mathbf{e}_{1:t}, \mathbf{h}_0) \tag{6.3}$$

The recurrent structure encodes short-term temporal dependencies by integrating past observations into the hidden state $\mathbf{h}_t$, initialised from $\mathbf{h}_0 = 0$. At each timestep, $\mathbf{h}_t$ summarises past information and is projected through a linear layer to produce frame logits:

$$\mathbf{z}_t = W_c\mathbf{h}_t + b_c \tag{6.4}$$

MiniROAD[21] achieves high throughput and low latency through its compact recurrent design, making it a widely used baseline for online OAD and a common recognition backbone in several PMD pipelines.
However, the limited temporal modelling capacity of the GRU hampers long-range consistency and the modelling of fine-grained hand–object interactions, often leading to unstable predictions and fragmented segments in egocentric videos.

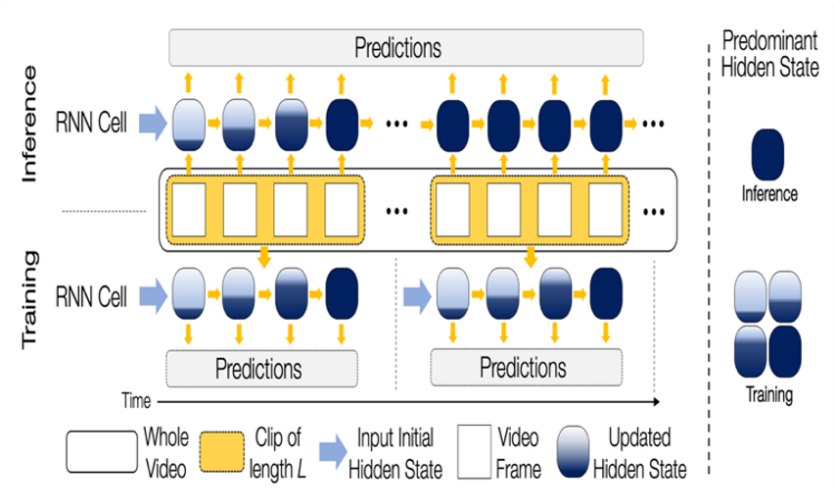


Figure 6.1: MiniROAD processes fixed length clips during training to produce per-frame predictions, while at inference it operates causally over the video stream, updating its hidden state to generate online frame logits

6.2 Agnostic OAD with Multimodal LLMs

As an initial exploratory study, we investigated a task-agnostic OAD approach based on **Multimodal Large Language models (MLLMs)** without task-specific training, aiming to assess whether a general-purpose multimodal model could provide meaningful recognition performance out of the box.

Experiments were conducted exclusively on Assembly101-O, where the video stream was divided into non-overlapping 2-second clips and each segment was independently analysed by the MLLM to predict the corresponding action. Recognition was formulated as a closed-set classification problem over the dataset action vocabulary, with predictions returned in a fixed JSON format to ensure consistent and reliable evaluation.

As shown in Table 6.2.1, when only raw video input and the predefined action vocabulary were provided in the prompt, all evaluated MLLMs exhibited limited recognition performance. Accuracy typically remained below 20%, with consistently low F1-scores across architectures. These results indicate that, despite their strong general visual and multimodal capabilities, the models struggle to capture procedural distinctions. In particular, they show limited sensitivity to subtle hand–object interactions and small variations in object configuration, which are critical for distinguishing between visually similar actions in egocentric assembly scenarios.

MLLM	FPS	mAP	mcAP	Acc	Prec	Rec	F1
Gemma3-12B[41]	0.40	1.40	62.83	9.19	0.90	2.11	1.26
Gemma3-27B[41]	0.93	1.52	62.71	6.17	2.84	1.50	1.96
InternVL2.5-8B[37]	10.55	1.48	62.82	7.08	1.99	2.05	2.02
Qwen2.5-VL-32B[36]	2.38	1.40	62.61	12.20	0.44	1.57	0.68
Qwen2.5-VL-72B[36]	2.13	1.54	62.75	13.40	1.12	1.88	1.43
VidiQA[42]	0.68	1.35	62.62	3.16	0.74	1.70	1.03
gemini2.0-flash[43]	8.54	2.25	63.99	15.51	3.34	4.51	3.82
gemini2.0-flash-thinking[43]	7.50	1.96	63.60	17.32	3.49	3.50	3.49
gemini2.5-flash[43]	3.19	3.14	63.70	19.43	7.22	3.77	4.95
gemini2.5-pro[43]	4.70	2.90	63.97	18.83	6.64	4.43	5.27
learnlm2.0-flash	3.56	1.84	63.33	16.27	3.33	3.00	3.15
Non-MLLM baseline							
Model	FPS	mAP	mcAP	Acc	Prec	Rec	F1
MiniROAD[21]	**11454.10**	**12.98**	**75.03**	**51.85**	**13.29**	**11.09**	**12.09**

Table 6.2.1: Agnostic MLLM OAD results on Assembly101-O compared with baseline

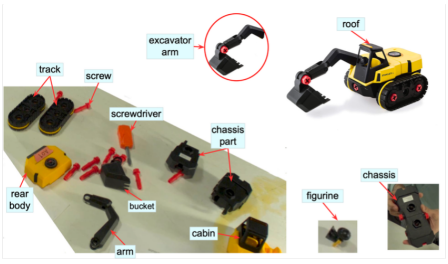


Figure 6.2: Example PDF used to support MLLM for OAD

To investigate whether recognition errors were primarily caused by weak object grounding, we augmented the prompt with a dataset provided PDF containing structured visual examples and labels of all toy components. This document offers explicit object context, allowing the model to associate textual action labels with concrete visual references and potentially reduce ambiguity between visually similar parts.

To quantify the impact of structured object context, we compared agnostic MLLM OAD performance with and without the dataset provided PDF. As reported in Table 6.2.2, models supporting external document conditioning exhibit consistent and substantial improvements across all recognition metrics.

Model	FPS	mAP	mcAP	Acc	Prec	Rec	F1
MLLM (no PDF context)							
gemini2.0-flash[43]	8.54	2.25	63.99	15.51	3.34	4.51	3.82
gemini2.5-flash[43]	3.19	3.14	63.70	19.43	7.22	3.77	4.95
gemini2.5-pro[43]	4.70	2.90	63.97	18.83	6.64	4.43	5.27
MLLM (with PDF context)							
gemini2.0-flash[43]	2.36	5.24	65.80	25.60	11.48	8.66	9.87
gemini2.5-flash[43]	0.78	6.37	66.47	26.51	11.04	10.37	10.69
gemini2.5-pro[43]	1.15	8.51	68.91	29.52	12.53	10.70	11.54
Task-trained baseline							
MiniROAD[21]	**11454.10**	**12.98**	**75.03**	**51.85**	**13.29**	**11.09**	**12.09**

Table 6.2.2: Agnostic MLLM OAD with and without PDF context, alongside the baseline

The improvements are systematic across architectures and not limited to a single configuration, with accuracy increasing from 15.51% to 25.60% for gemini2.0-flash, from 19.43% to 26.51% for gemini2.5-flash, and from 18.83% to 29.52% for gemini2.5-pro. This suggests that the primary limitation of agnostic MLLMs lies in insufficient object grounding rather than general visual reasoning capability.
By providing aligned visual references through the PDF, the models more effectively associate textual action labels with concrete object instances, reducing ambiguity between visually similar components.

Despite these gains, even the best performing configuration remains clearly inferior to the task trained MiniROAD[21] baseline. This limitation of agnostic MLLMs is consistent with findings from InstructionBench[44], where even the best performing multimodal models remain clearly inferior to task trained baselines in instructional video understanding. Moreover, using API-based inference introduces significant computational overhead and latency, making them unsuitable for strict online action detection requirements. For these reasons, MLLM recognition is not adopted in the subsequent stages of the pipeline.

6.3 Online Graph-Temporal Network (OGTN)

Online Graph-Temporal Network (OGTN) is a novel recogniser proposed in this thesis designed to balance accuracy, temporal stability, and efficiency under strict online constraints. The architecture is structured around three core design principles: an object-centric abstraction realised through latent slots without explicit object detection, intra-frame relational reasoning enabled by a compact attention graph encoder, and temporal modelling introduced only after feature compression to maintain efficiency.

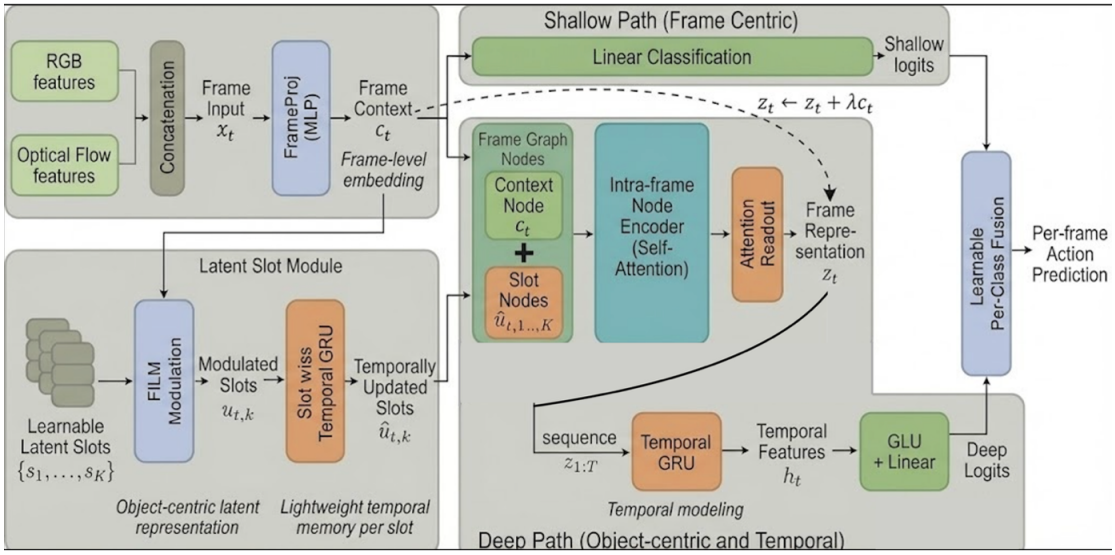


Figure 6.3: Architecture of OGTN for OAD. The frame context token c_t supports a frame-level shallow classifier, while the deep path builds object-centric representations via FiLM-modulated latent slots, performs intra-frame relational reasoning with self-attention, and injects temporal information through lightweight GRU modules. A learnable per-class fusion balances instantaneous and temporally enriched evidence to produce per-frame logits.

At each time step t, the model receives pre-extracted RGB and optical-flow embeddings, in direct analogy with the MiniROAD baseline, which are concatenated and projected through a lightweight multi-layer perceptron MLP (FrameProj) to obtain a compact frame level context embedding

$$\mathbf{c}_t = \text{FrameProj}(\mathbf{x}_t) \tag{6.5}$$

The embedding c_t provides a global summary of the current frame and branches into two complementary paths: a shallow path that produces predictions based solely on the current frame, and a deep path for structured object-centric and temporal modelling. In the shallow branch, c_t supports decisions based on current frame cues, while in the deep branch it conditions latent representations and is

injected, via residual connections, into temporally enriched features. This dual use preserves strong appearance information while introducing temporal context only when necessary.

Shallow Path

The shallow path directly maps $\mathbf{c}_t$ to logits through a linear transformation:

$$\ell_t^{shallow} = W_s \mathbf{c}_t + b_s \tag{6.6}$$

This pathway bypasses any form of temporal modelling or structured reasoning, providing responsive predictions based solely on the current frame.
It is particularly effective for action classes that are visually distinctive and can be reliably recognised from a single observation.

Deep Path

In parallel to the shallow branch, the deep path constructs a richer representation by introducing an implicit object-centric abstraction based on a small set of learnable latent slots $\{\mathbf{s}_1, \ldots, \mathbf{s}_K\}$, with $\mathbf{s}_k \in \mathbb{R}^d$. These slots are shared across the dataset and do not correspond to predefined object categories. Instead, they form a compact and flexible representation space that captures recurring entities, object parts, or interaction-relevant regions that emerge consistently across procedural sequences.
To adapt the latent slots to the current frame, a FiLM-style modulation conditioned on the frame context embedding $\mathbf{c}_t$ is applied:

$$[\gamma_t; \beta_t] = W_f c_t + b_f \quad \Rightarrow \quad u_{t,k} = \gamma_t \odot s_k + \beta_t \tag{6.7}$$

This modulation dynamically specialises each slot based on the global visual context, enabling the same latent prototype to represent different scene aspects over time.
In order to introduce short-term temporal persistence before structural reasoning, each slot stream is equipped with a lightweight slot-wise GRU:

$$\tilde{\mathbf{u}}_{1:T,k} = \mathrm{GRU}(\mathbf{u}_{1:T,k}) \tag{6.8}$$

Since the number of slots K is small, this recurrence introduces temporal smoothing at minimal computational cost, reducing frame level noise while preserving efficiency.
For each frame, a compact set of graph nodes is constructed by concatenating the frame context embedding and the temporally updated slots:

$$\mathbf{V}_t = [\mathbf{c}_t; \tilde{\mathbf{u}}_{t,1}; \ldots; \tilde{\mathbf{u}}_{t,K}] \tag{6.9}$$

This graph models structural relationships within a single frame, capturing interactions between latent object-centric entities and the global context. Importantly, no temporal state is stored in the graph itself: each frame is processed independently, allowing efficient parallel computation.

Intra-frame reasoning is performed by a lightweight attention graph encoder, implemented with multi-head self-attention and residual feed-forward blocks:

$$\mathbf{V}'_t = \mathrm{GNN}(\mathbf{V}_t) \tag{6.10}$$

The updated node representations are collapsed into a single frame descriptor through an attention readout. The resulting representation is then combined with the original frame context via residual injection:

$$\mathbf{z}_t \leftarrow \mathbf{z}_t + \lambda \mathbf{c}_t \tag{6.11}$$

This ensures that strong global appearance cues are preserved alongside structured relational information. Global temporal dependencies are introduced only after this pooling stage. The sequence of compact frame descriptors is processed by a temporal GRU:

$$\mathbf{h}_{1:T} = \mathrm{GRU}(\mathbf{z}_{1:T}) \tag{6.12}$$

By applying temporal modelling exclusively to compressed frame-level representations, the model captures action dynamics while avoiding the high computational cost of recurrent processing within the graph encoder or across large per-slot structures.

Per-Class Fusion

The final prediction is obtained by combining shallow and deep logits through a learnable per-class fusion mechanism. A vector of mixing coefficients $\boldsymbol{\alpha} \in [0,1]^{|\mathcal{Y}|}$ modulates the contribution of each path.

$$\ell_t = \alpha \odot \ell_t^{deep} + (1 - \alpha) \odot \ell_t^{shallow} \tag{6.13}$$

This formulation allows the model to learn how much temporal reasoning is required for each action class. Actions that are visually distinctive can rely primarily on instantaneous frame level evidence, while actions characterised by motion patterns or temporal structure benefit from the deep path.

Although Online Graph-Temporal Network (OGTN) may appear architecturally complex, it is designed for efficiency. Object-centric reasoning is performed through a small set of latent slots, avoiding explicit object detection, while intra-frame interactions are captured by attention over a limited number of nodes. Temporal modelling is introduced only where necessary: slot-wise recurrence provides local stability, and global dependencies are modelled through a lightweight temporal GRU on pooled frame features. Finally, dual-path classification combines frame-level cues from the shallow branch with temporally enriched representations from the deep branch through learnable per-class fusion, enabling a strong trade-off between accuracy, temporal stability, and throughput for online action recognition.

6.4 Online Temporal Transformer (OTT)

Online Temporal Transformer (OTT) is another recogniser proposed in this thesis, designed to model procedural dynamics through a fully attention-based temporal architecture while strictly respecting online inference constraints. The model follows a deliberately simple design: each frame is first mapped to a compact representation, and all temporal reasoning is delegated to a causal Transformer operating over the resulting sequence.

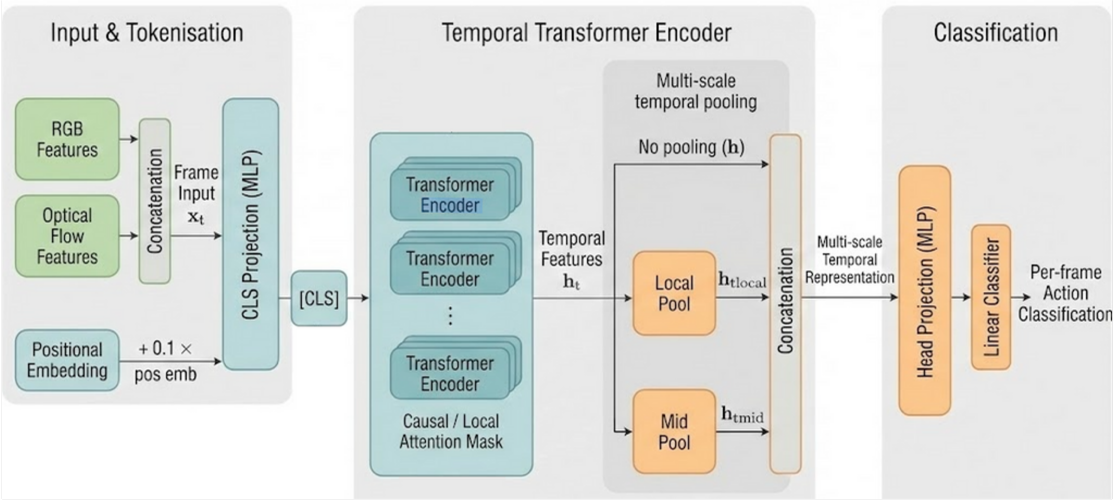


Figure 6.4: Workflow of OTT for OAD. Frame tokens are processed by a causal temporal Transformer and enhanced through multi-scale pooling before per-frame classification.

As in the other recognisers in this chapter, OTT operates on pre-extracted visual features. At each time step t, RGB and optical-flow embeddings are concatenated to form the frame input. Each frame is summarised into a single CLS-like token through a lightweight projection block, which can be interpreted as a shallow MLP composed of normalisation, linear projection, non-linearity, and dropout

$$\mathbf{c}_t = \text{Dropout}(\text{GELU}(\text{LayerNorm}(W\mathbf{x}_t + b))) \tag{6.14}$$

To encode temporal order, a positional embedding is added to each frame token

$$\mathbf{c}_t \leftarrow \mathbf{c}_t + 0.1\,\mathbf{p}_t \tag{6.15}$$

where $\mathbf{p}_t$ is a learnable embedding encoding the relative position of frame t within the sequence, rather than physical time. Learning $\mathbf{p}_t$ allows the model to adapt temporal ordering to the non-uniform durations and transition patterns typical of procedural data. The scaling factor ensures that positional information remains auxiliary, while visual evidence dominates the representation.

The frame embeddings $\mathbf{c}_{1:T} \in \mathbb{R}^{T\times d}$ are processed as a temporal sequence by a Transformer operating along the time dimension. Temporal modelling is performed by stacking Transformer layers that update all time steps in parallel while enforcing a causal attention constraint. For each time step t, the output representation is computed as

$$\mathbf{h}_t = \sum_j \alpha_{t,j}\mathbf{v}_j, \qquad \alpha_{t,j} = \text{softmax}(\mathbf{q}_t \cdot \mathbf{k}_j), \qquad t - w + 1 \leq j \leq t \tag{6.16}$$

where $\mathbf{q}_t$ is the query derived from the current frame representation $\mathbf{c}_t$, and $\mathbf{k}_j$, $\mathbf{v}_j$ are keys and values obtained via learned linear projections at time j. The causal window of size w ensures that each $\mathbf{h}_t$ depends only on past and present frames. Despite selective, content-aware aggregation, the Transformer's frame outputs may remain locally unstable, so OTT enhances temporal stability and action persistence through multi-scale causal average pooling.
Given the sequence $\mathbf{h}_{1:T}$, a pooled representation with kernel size k is defined as

$$h_t^{(k)} = \frac{1}{k}\sum_{i=t-k+1}^{t} h_i \tag{6.17}$$

By instantiating this operation with different kernel sizes, the model constructs short-term and mid-term temporal representations that are concatenated with the Transformer output before projection and classification.

$$\mathbf{z}_t = \text{MLP}([\mathbf{h}_t; \mathbf{h}_t^{short}; \mathbf{h}_t^{long}]) \tag{6.18}$$

Finally, a linear classifier produces per-frame action logits

$$\ell_t = W_c\mathbf{z}_t + b_c \tag{6.19}$$

Online Temporal Transformer (OTT) models temporal dependencies for online action recognition by processing frame embeddings through a causal Transformer with a fixed attention window. This design ensures that predictions at time t depend only on past and present frames, enabling strictly online inference. Learnable positional embeddings capture the temporal ordering of procedural actions, while windowed causal attention limits the temporal receptive field and keeps computation efficient. To further stabilise predictions, OTT augments the Transformer output with multi-scale causal average pooling, providing short- and mid-term temporal context before classification. This design yields a compact and effective baseline for online action recognition.

6.5 Results: Online Action Detection

This section reports the Online Action Detection (OAD) models evaluated for the recognition component of this thesis, assessing both prediction quality and computational throughput (FPS) under strict online constraints.

	Assembly101-O							EpicTent-O						
Model	**FPS**	**Acc**	**Prec**	**Rec**	**F1 (%)**	**mAP**	**mcAP**	**FPS**	**Acc**	**Prec**	**Rec**	**F1 (%)**	**mAP**	**mcAP**
MLLM no PDF														
gemini2.0-flash[43]	8.54	15.51	3.34	4.51	3.82	2.25	63.99	–	–	–	–	–	–	–
gemini2.5-flash[43]	3.19	19.43	7.22	3.77	4.95	3.14	63.70	–	–	–	–	–	–	–
gemini2.5-pro[43]	4.70	18.83	6.64	4.43	5.27	2.90	63.97	–	–	–	–	–	–	–
MLLM with PDF														
gemini2.0-flash[43]	2.36	25.60	11.48	8.66	9.87	5.24	65.80	–	–	–	–	–	–	–
gemini2.5-flash[43]	0.78	26.51	11.04	10.37	10.69	6.37	66.47	–	–	–	–	–	–	–
gemini2.5-pro[43]	1.15	29.52	12.53	10.70	11.54	8.51	68.91	–	–	–	–	–	–	–
Non-MLLM														
MiniROAD[21]	11454.10	51.85	**13.29**	11.09	12.09	12.98	75.03	12062.20	59.56	34.98	34.36	34.67	37.17	**82.18**
OTT⋆	12922.15	**52.22**	11.49	**12.22**	11.84	12.32	74.03	**15581.21**	58.00	34.27	30.89	32.49	33.61	78.62
OGTN⋆	**15411.95**	52.14	13.19	11.17	**12.10**	**13.10**	**76.01**	14740.02	**62.32**	**36.18**	**36.93**	**36.55**	**38.99**	80.92

⋆ Models proposed in this thesis

Table 6.5.1: Comparison OAD performance on Assembly101-O and EpicTent-O, evaluating both recognition quality and throughput (FPS)

Table 6.5.1 shows that agnostic MLLM approaches are unsuitable for OAD, consistent with prior work[44]. Even with PDF-based object grounding, they remain significantly less accurate than trained models and too slow for online deployment.

MiniROAD[21] achieves high throughput (over 10k FPS on both datasets), confirming the efficiency of its lightweight GRU design. However, its limited temporal capacity reduces robustness in fine-grained and visually similar procedural steps, with constrained precision and recall in longer action sequences.

Online Temporal Transformer (OTT) introduces causal self-attention over compact frame embeddings, improving temporal modelling while preserving very high throughput. On Assembly101-O it slightly improves accuracy and recall compared to MiniROAD[21], while maintaining extremely efficient inference and reaching the highest FPS on EpicTent-O.

Online Graph-Temporal Network (OGTN) delivers the strongest overall performance by combining object abstraction, lightweight relational reasoning, and efficient temporal modelling. On Assembly101-O, it improves over MiniROAD[21] by +0.3% Accuracy and +0.15% mAP/mcAP while increasing throughput by +34.6%. On EpicTent-O, the gains are more pronounced (+2.76% Accuracy, +1.88% F1, +1.82% mAP, +22.2% FPS), highlighting its robustness in more complex settings.

Overall, OGTN provides the best balance of accuracy, temporal consistency, and efficiency, and is selected as the main recogniser for subsequent modules.

Chapter 7

Frame to Step Aggregation

Frame-wise predictions from the OAD module are inherently noisy, especially around step transitions, where oscillations are frequent. In the proposed pipeline, this stream must be converted into a **procedural step sequence** consumed by anticipation module and mistake detection. Aggregation determines how many consecutive predictions are required before committing a final step label, thus directly controlling commit latency. The objective is to **minimise commit latency** while maintaining temporal stability and structural coherence.
The aggregation strategies are measured using **Levenshtein similarity**, which accounts for step insertions, deletions, and substitutions, and thus jointly reflects segmentation stability and step ordering consistency.
Given two step sequences A and B, the similarity is defined as:

$$\mathrm{LevSim}(A, B) = 1 - \frac{d_{\mathrm{lev}}(A, B)}{\max(\|A\|, \|B\|)} \tag{7.1}$$

The search is driven by a strict latency constraint: **commit latency must remain below 4 seconds**, improving over the baseline non-overlapping 200 frame aggregation (≈ 6.67 seconds). The goal is to reduce latency while preserving step consistency and correct ordering.
All strategies are strictly causal: at time t, decisions depend only on current and past frames. Even for block based methods, labels are committed only after the last frame of the block is observed, ensuring full compatibility with online deployment.

7.1 Label based aggregators

We start with aggregation strategies that operate directly on frame predictions. These model agnostic rules over the label stream and provide an explicit control over commit latency, serving as a clear reference point to study the latency–quality trade-off in online settings.

7.1.1 Non-overlapping Majority Aggregation (NOMA)

NOMA divides the frame sequence into non-overlapping blocks of size B. Each block is assigned a single step label through majority voting over its frames. While this ensures strong temporal stabilisation, the commit latency is directly proportional to the block size.

$$y_{[s-B+1,\,s]} = \arg\max_{c} \sum_{i=s-B+1}^{s} \mathbf{1}[y_i = c] \tag{7.2}$$

7.1.2 Minimum Duration Filter + NOMA (MDF+NOMA)

This strategy introduces a minimum duration constraint before block aggregation, validating a step transition only if the new label remains stable for a fixed number of consecutive frames. Short lived fluctuations are filtered at the frame level, and then NOMA is applied to the stabilised sequence.
Formally, a transition from label a to b at time t is accepted only if:

$$\sum_{i=t-L+1}^{t} \mathbf{1}[y_i = b] = L \quad \Rightarrow \quad \text{apply NOMA}$$

7.1.3 Hysteresis (HYST)

Hysteresis enforces temporal inertia by delaying step transitions until sufficient evidence for a new label has accumulated. Let s_t denote the currently committed step label at time t, and let y_t be the frame prediction. The filter maintains a current state s, a candidate state c, and a persistence counter k. At each time step:

$$k = \begin{cases} k+1 & \text{if } y_t = c, \\ 1 & \text{if } y_t \neq c. \end{cases}$$

A state transition is accepted only when the candidate label persists for at least τ consecutive frames, i.e., when $k \geq \tau$, the current state is updated as $s \leftarrow c$.
Until then, the output remains unchanged, reducing short lived oscillations.

7.1.4 Overlapping Mode Aggregation (OMA)

Overlapping Mode Aggregation (OMA) combines a trailing causal voting window with a fixed commit schedule. Instead of using disjoint blocks, it updates the committed label at regular intervals, while each decision leverages a longer past context. Formally, for a chunk ending at frame t_{end}, the committed label is:

$$y_{\text{chunk}} = \arg\max_{c} \sum_{i=\max(1,t_{\text{end}}-W+1)}^{t_{\text{end}}} \mathbf{1}[y_i = c] \tag{7.3}$$

7.2 Logit based aggregators

Label based aggregation operates on discrete predictions and so ignores the confidence information available in the recogniser outputs. To leverage this signal, we also consider aggregation strategies that work directly on the frame logits $\ell_t \in \mathbb{R}^C$. By accumulating soft evidence over time, these methods can distinguish between confident and uncertain predictions and reduce instability around step boundaries.

7.2.1 Soft Overlapping Mode Aggregation (Soft-OMA)

Soft-OMA operates directly on the recogniser's logits, accumulating probabilistic evidence instead of using majority voting. This allows step decisions to reflect both temporal consistency and model confidence, improving stability near uncertain step boundaries. As in OMA, labels are committed every fixed number of frames using a strictly trailing causal window W_t. Class evidence is aggregated by averaging log-probabilities over the trailing window W_t, and the committed label corresponds to the class with maximum aggregated confidence. Formally:

$$\hat{y}_{\text{chunk}} = \arg\max_c \frac{1}{|W_t|} \sum_{i \in W_t} \log p_i(c), \quad with \quad p_t(c) = \frac{\exp(\ell_t(c))}{\sum_j \exp(\ell_t(j))} \tag{7.4}$$

7.2.2 Exponential Moving Average (EMA) Aggregation

EMA aggregation introduces temporal smoothing directly in probability space. At each frame, logits are converted into class probabilities and used to update a running estimate via an exponential moving average. The current probabilities are combined with the previous smoothed estimate $\bar{p}_{t-1}$, allowing changes in class confidence to be integrated gradually over time.

$$\bar{p}_t = (1-\gamma)\bar{p}_{t-1} + \gamma p_t = \gamma p_t + \gamma(1-\gamma)p_{t-1} + \gamma(1-\gamma)^2 p_{t-2} + \dots \tag{7.5}$$

The smoothing factor $\gamma \in (0,1]$ controls adaptation speed: smaller values increase temporal smoothing, while larger ones enable faster reactions.
Step decisions are taken only at chunk boundaries, where the label with the highest smoothed probability is assigned to the whole chunk.

$$y_{\text{chunk}} = \arg\max_c \bar{p}_{t_{\text{end}}}(c) \tag{7.6}$$

7.2.3 Viterbi Aggregation with Transition Penalty (VTP)

This strategy enforces temporal consistency by discouraging frequent step changes while operating on confidence scores. It accumulates evidence over time and favours staying in the current step unless strong support for a transition emerges.
At each frame t, logits ℓ_t are converted to probabilities $p_t(c)$ via softmax, and their logarithm is used as frame-level evidence:

$$e_t(c) = \log p_t(c) \quad with \quad p_t(c) = \frac{\exp(\ell_t(c))}{\sum_j \exp(\ell_t(j))}. \tag{7.7}$$

For each class c, an accumulated score $\delta_t(c)$ is maintained. This score represents the strength of the evidence for being in step c at time t, taking past frames into account. The update rule is:

$$\delta_t(c) = e_t(c) + \max\left(\delta_{t-1}(c), \max_{j \neq c} \delta_{t-1}(j) - \lambda\right) \tag{7.8}$$

where $\lambda \geq 0$ is a transition penalty. This introduces a stay bias: remaining in the same step has no cost, while switching is allowed only if the accumulated evidence overcomes λ. Consequently, short-lived fluctuations do not trigger transitions, whereas consistent evidence leads to stable changes.
Step decisions are committed at chunk boundaries. Let t_{end} be the last frame of a chunk; the assigned label is:

$$y_{\text{chunk}} = \arg\max_c \delta_{t_{\text{end}}}(c) \tag{7.9}$$

7.3 Results Aggregation Methods

Frame predictions tend to fluctuate, whereas procedural monitoring requires a stable step sequence. The **frame-to-step aggregation** module groups consecutive predictions and commits a step only once enough temporal evidence has accumulated. As aggregation produces a symbolic step sequence instead of frame outputs, frame metrics are not appropriate. We therefore use **Levenshtein Similarity**, which evaluates how many insertions, deletions, and substitutions are needed to transform the predicted sequence into the ground-truth one.

$$\mathrm{LevSim}(A,B) = 1 - \frac{d_{\mathrm{Lev}}(A,B)}{\max(\|A\|,\|B\|)}, \quad \mathrm{LevSim}(A,B) \in [0,1] \tag{7.10}$$

where $d_{\mathrm{Lev}}(A,B)$ denotes the minimum number of insertions, deletions, or substitutions required to transform sequence A into B.

The state-of-the-art baseline adopts the NOMA (Non-Overlapping Mode Aggregation) strategy with fixed blocks of approximately **200 frames** ($\approx 6.67\,\mathrm{s}$).
Under this setting, it achieves a Levenshtein Similarity of **34.62** on Assembly101-O and **43.24** on EpicTent-O.

Our objective is to attain comparable structural stability while reducing aggregation latency to $\leq 4\,\mathbf{s}$, minimizing the aggregation delay while preserving equivalent structural performance. Latency is computed assuming 30 FPS input videos (30 frames = 1 second), and all delays are reported accordingly.

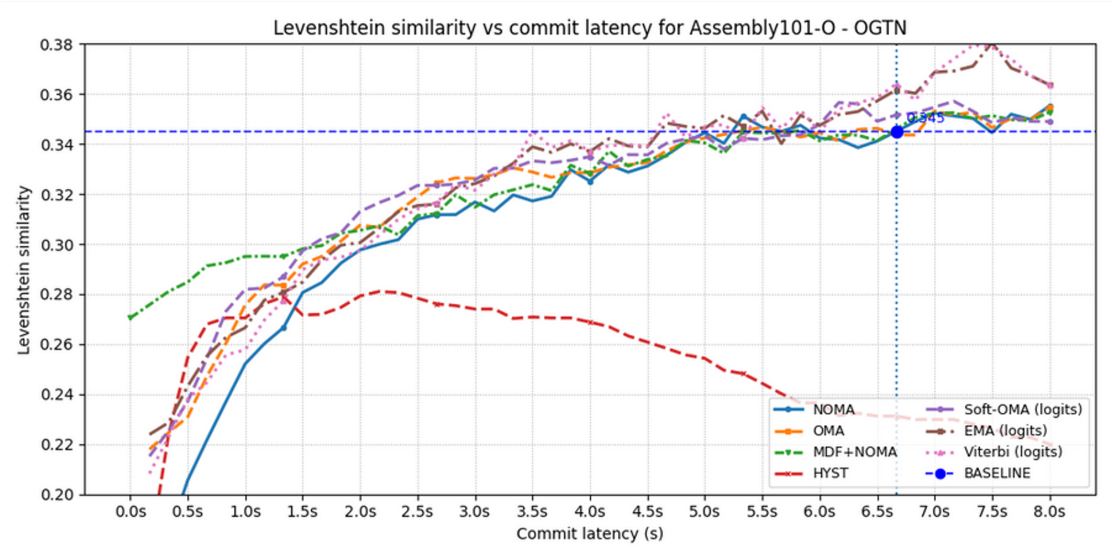


Figure 7.1: Levenshtein Similarity achieved by different aggregation strategies as a function of commit latency on Assembly101-O

As shown in Figure 7.1, increasing the aggregation window generally improves Levenshtein Similarity, as longer temporal context enhances structural stability, but at the cost of higher commit latency. Label based methods benefit from longer context but operate on discrete class predictions, ignoring confidence information, whereas logit based strategies exploit probability scores to achieve stronger agreement at comparable latency.

	Assembly101-O							**EpicTent-O**						
Agg Method	≤2s	≤3s	≤3.5s	≤4s	≤5s	≤6s	>6s	≤2s	≤3s	≤3.5s	≤4s	≤5s	≤6s	>6s
Label Based														
NOMA	29.76	31.68	31.96	32.97	33.95	34.42	35.52	28.89	35.41	36.64	37.95	41.63	42.06	46.22
MDF+NOMA	30.55	31.97	32.37	33.15	34.04	34.57	35.27	34.33	38.26	38.94	40.63	42.85	45.00	**47.59**
HYST	27.92	28.10	28.10	28.10	28.10	28.10	23.23	**41.77**	**45.03**	**47.24**	**49.00**	**50.64**	**50.64**	46.44
OMA	30.76	32.64	33.05	33.05	34.24	34.42	35.46	35.50	39.45	41.92	42.70	42.70	43.22	46.46
Logits Based														
Soft-OMA	31.30	32.56	33.32	33.49	34.33	34.78	35.71	36.35	40.40	41.97	42.31	43.51	46.53	47.04
EMA	30.06	32.41	33.89	34.01	34.83	35.31	**38.05**	29.62	33.85	35.37	36.97	40.11	42.95	46.23
VTP	29.74	32.36	**34.52**	**34.52**	**35.24**	**35.46**	37.95	27.96	33.09	33.80	36.04	41.07	41.76	46.14

Table 7.3.1: Levenshtein Similarity for OGTN across aggregation strategy under different commit latency thresholds

The results in Table 7.3.1 show different behaviours across the two datasets when compared to the NOMA-200 baseline, which reaches a Levenshtein Similarity of 34.62 on Assembly101-O and 43.24 on EpicTent-O with a commit latency of 6.67s.

On **Assembly101-O**, performance increases steadily with latency, but it is clear that methods exploiting confidence scores reach the same level of structural agreement with shorter commit times. For instance, VTP attains 34.52 already at ≤3.5s, while most label driven strategies require longer windows to approach similar values. This shows that leveraging probability information reduces the need for extended temporal accumulation.

On **EpicTent-O**, the behaviour is less uniform. While confidence strategies improve with latency, the strongest results in the low- and mid-latency range are achieved by HYST, which peaks at 50.64 at ≤5s–≤6s. Nonetheless, across short latency thresholds, approaches using logits information generally outperform purely label based strategies at comparable or lower commit times.

The final objective was to reduce aggregation latency to $\leq 4\,\mathrm{s}$ (compared to the 6.67 s from baseline) while preserving a Levenshtein Similarity comparable to NOMA with 200-frame blocks. Latency is computed assuming 30 FPS input videos (30 frames = 1 second), and all delays are reported accordingly.

	Assembly101-O				EpicTent-O			
Model	**Type Agg**	**Frame Lat**	**Delay Agg**	**LevSim**	**Type Agg**	**Frame Lat**	**Delay Agg**	**LevSim**
MiniROAD[21]	noma 200	200f	6.67s	34.62	noma 200	200f	6.67s	43.24
OTT	softoma W240 s120	120f	4.0s	34.07	hyst tau 115	115f	3.83s	36.17
OGTN	viterbi s105 pen1	**105f**	**3.34s**	34.52	hyst tau 105	**105f**	**3.34s**	**47.24**

Table 7.3.2: Selected low-latency ($\leq$ 4s) aggregation configurations

Table 7.3.2 summarizes the configurations that satisfy the $\leq$ 4s latency constraint while maintaining structural agreement with the baseline. The results show that the latency–quality trade-off can be shifted substantially without degrading sequence consistency.

On Assembly101-O, both OTT and OGTN reach Levenshtein Similarity values comparable to the 200 frame NOMA baseline despite operating with nearly half the aggregation delay. Notably, the OGTN configuration based on Viterbi strategy (105 frames) achieves 34.52 LevSim at 3.34s, effectively preserving baseline structural alignment with significantly reduced commitment latency.

On EpicTent-O, hysteresis aggregation emerges as the most effective strategy.
The OGTN configuration with $\tau = 105$ not only satisfies the low latency constraint (3.34s) but surpasses the baseline in structural agreement (47.24 vs 43.24).
This indicates that, for this dataset, temporal consistency can be enforced earlier without relying on long fixed blocks.

These selected aggregation strategies for OTT and OGTN are subsequently integrated into the anticipation module, where the stabilised step sequence forms the input for next-step prediction. By combining low aggregation latency with strong structural coherence, the anticipation stage operates on representations that are both timely and procedurally reliable.

Chapter 8

Step Anticipation Module

The anticipation module is a core component of the procedural mistake detection (PMD) pipeline, operating after frame recognition and temporal aggregation. While these stages provide a stable symbolic description of the current step, the anticipation module predicts the next valid procedural action, enabling explicit reasoning about execution correctness. Within this framework, mistake detection is formulated as a consistency check between recognition and anticipation. A mistake occurs when the recognised step does not match the expected next step according to the procedural model. Formally, at time t, the mistake indicator is defined as:

$$\text{Mistake}(t) = \begin{cases} 1 & \text{if } \hat{y}^t_{\text{rec}} \neq \hat{y}^t_{\text{ant}} \\ 0 & \text{otherwise} \end{cases} \tag{8.1}$$

where $\hat{y}^t_{\text{rec}}$ denotes the step predicted by the recognition branch at time t, and $\hat{y}^t_{\text{ant}}$ denotes the next step prediction produced by the anticipation module given the observed procedural history up to time t.

Unlike the visual recognition module, which maps perceptual input to action labels, the anticipation module operates at the symbolic level. It models the procedural dynamics governing ordered action sequences, learning transition regularities and structural constraints over discrete step to predict the most plausible next action.

In this chapter, we compare two complementary anticipation paradigms: one based on Large Language Models, specifically Llama 3.2 (1B and 3B), used without fine tuning and relying on in-context reasoning, and another based on task trained models explicitly optimized on annotated procedural sequences to capture dataset transition dynamics.

Both paradigms are evaluated on ground-truth sequences and on OAD predicted sequences, capturing intrinsic prediction ability and realistic deployment under noisy inputs.

8.1 Step Anticipation with LLM

Large Language Models (LLMs) are well suited for step anticipation due to their ability to model structured sequences and procedural regularities through in-context reasoning, without requiring task-specific training. In this setting, anticipation is formulated as a symbolic sequence prediction problem, where the objective is to infer the most plausible next step given a partial procedural history.

The main motivation for using non-trained LLMs is their flexibility. Unlike trained anticipators tied to specific datasets, pretrained LLMs encode general procedural priors that enable adaptation to different procedures without retraining, which is particularly advantageous in online mistake detection where data may be limited. Crucially, when guided by a carefully designed prompt and provided with examples of correct step sequences extracted from the training set, LLMs can achieve meaningful anticipation performance despite the absence of explicit training. These in-context examples supply lightweight procedural grounding, enabling the model to infer plausible next steps by analogy and to capture ordering constraints and goal-driven dependencies.

In this thesis, step anticipation is investigated primarily using Llama 3.2 models with 3B and 1B parameters, showing that even relatively compact LLMs can deliver competitive performance when appropriately prompted and contextualised. This is particularly relevant for online settings, as it demonstrates that effective LLM anticipation does not necessarily require large or computationally expensive models.

The PREGO[10] and TI-PREGO[11] framework previously demonstrated that LLM next step prediction can provide effective supervisory signals for online mistake detection. Building on this evidence, we adopt LLMs as zero-/few-shot anticipators, providing a flexible and training-free alternative to task-specific models, whose trade-offs are examined in the following sections.

8.1.1 Prompting Strategies for Procedural Anticipation with LLM

In the proposed formulation, procedural histories are encoded as symbolic sequences of discrete action identifiers. Given a partial sequence, the model is required to output a single next symbol corresponding to the expected procedural step.
This strict constraint ensures compatibility with the online mistake detection pipeline and prevents uncontrolled verbosity.

All prompts proposed in this thesis follow a consistent structure. They start with a task description defining the model's role as procedural sequence completion, followed by in-context demonstrations extracted from training data. Each example includes a symbolic action sequence and its correct next step, illustrating valid transitions and implicitly defining the action vocabulary for the LLM. The prompt then asks for the next action of a new sequence prefix, enabling next-step prediction without training.

Methodologically, the prompts adopt an **Assisted Chain-of-Thought (ACoT)** style, encouraging internal structured reasoning (e.g., hypothesis generation and self-verification) while constraining the final output to a single symbolic action. Within this framework, four variants are explored, each imposing different reasoning constraints and selection strategies:

1. **Direct Procedural Completion (DPC):** The simplest formulation, where the model predicts the next step solely based on in-context examples.
   No explicit verification is enforced; the LLM acts as a pattern-completion mechanism and serves as a stable baseline.
2. **Self-Consistency via Dual Internal Experts (SC):** The model internally generates multiple candidate independent steps and selects the final prediction by enforcing consistency with the demonstrated transitions, promoting internal validation.
3. **Two-Pass Refinement with Verification (2PV):** The model first produces an initial guess and then verifies and possibly revises it, focusing on error correction rather than agreement between parallel hypotheses.
4. **Multi-Hypothesis Coherence-Based Selection (MHC):** The model considers multiple possible next steps and selects the one yielding the most coherent procedural continuation, encouraging local lookahead and global consistency.

Overall, the prompts range from simple completion to more structured ACoT-style reasoning. All variants preserve the same external behaviour, predicting a single next-action symbol, so that performance differences reflect reasoning constraints rather than input or output changes.

8.2 Step Anticipation with Trained Models

In addition to LLM anticipation, this thesis considers task trained step anticipators, which are explicitly optimised on annotated procedural sequences.
In this section, anticipation is treated as a supervised next-step prediction problem, where the model learns dataset-specific transition dynamics directly from ground-truth data.

Compared to LLMs, task trained anticipators are less flexible under distribution shifts, such as unseen procedures or actions. However, by being directly optimized on target data, they typically achieve higher in-domain accuracy, learning precise transition patterns instead of relying on general priors. In addition, they are lightweight, have predictable computational cost, and provide significantly lower inference latency, making them well suited for real-time online mistake detection.

In the following section, we describe the architecture and training setup of the proposed task trained step anticipator and evaluate its performance under both ideal conditions, using ground-truth step sequences and realistic online conditions, where anticipation operates on OAD predicted sequences.

8.2.1 Loss Function for Trained Step Anticipation

Training a step anticipator on procedural datasets involves strong class imbalance, as some steps occur frequently while others appear rarely. To mitigate this, we adopt a class balanced cross-entropy loss based on the Effective Number of Samples formulation. Instead of weighting classes purely by their raw frequency, this approach reduces the impact of additional samples for already frequent classes.
As a result, rare classes receive higher weights, while the weights of frequent classes decrease more gradually as the number of samples n_c grows. This produces a smoother and stable rebalancing compared to standard inverse-frequency weighting.
Let n_c denote the number of training samples for class c, with class weights computed as:

$$w_c = \frac{1-\beta}{1-\beta^{n_c}} \qquad \text{where } \beta \in [0,1) \tag{8.2}$$

These weights are incorporated into a weighted cross-entropy objective:

$$\mathcal{L}_{\text{CB-CE}} = -\sum_c w_c \, y_c \log p_c \tag{8.3}$$

The use of class balanced cross-entropy encourages the model to allocate learning capacity more evenly across procedural steps, improving anticipation accuracy for infrequent actions without sacrificing performance on common ones.
This is particularly important in the context of mistake detection, where rare steps often correspond to critical transitions whose misprediction can lead to false negatives or delayed error detection.

8.2.2 Recurrent Neural Network for Step Anticipation

As a first task trained baseline for step anticipation, we employ a recurrent language model operating on symbolic action sequences. Given a variable length history of previously observed steps, the model predicts the next procedural action in a fully causal manner. Each action symbol is first transformed into a continuous, learnable vector representation through an embedding layer to encode discrete procedural steps into a dense space, enabling the recurrent model to capture semantic and structural relationships between actions before temporal processing. The resulting sequence of embeddings is processed by a recurrent neural network, implemented in this work using a **multi-layer Gated Recurrent Unit (GRU)**:

$$\mathbf{h}_{1:t} = \mathrm{GRU}(\mathbf{x}_{1:t}) \tag{8.4}$$

Through its recurrent dynamics, the network progressively aggregates information from past actions and builds an internal representation of the procedural context. The hidden representation corresponding to the final time step $\mathbf{h}_t \in \mathbb{R}^d$ is interpreted as a summary of the observed action history and is passed through a linear prediction head, producing logits over the action vocabulary from which the next step is predicted:

$$\mathbf{z} = W\mathbf{h}_t + \mathbf{b} \tag{8.5}$$

This recurrent anticipator combines low inference latency, strict causality, and direct learning of step-to-step transition dynamics. While its representational capacity is more limited than that of larger sequence models, it provides strong performance within the training distribution and serves as an efficient and reliable baseline for comparison with both LLM anticipation and more expressive trained architectures.

8.2.3 Causal Transformer with RoPE (CRAT)

To increase modelling capacity beyond recurrent baselines, we adopt a causal Transformer model for step anticipation. The resulting architecture, referred to as the **Causal RoPE Anticipation Transformer (CRAT)**, operates on symbolic action sequences and is explicitly designed for fully causal next-step prediction. Discrete action identifiers are first mapped to continuous embeddings and then processed by a stack of Transformer blocks with multi-head self-attention and position-wise feed-forward layers. Strict causality is enforced through a triangular mask, ensuring that each position attends only to past and present steps:

$$\mathbf{h}_{1:T} = \text{softmax}\left(\frac{Q'K'^{\top}}{\sqrt{d}} + M_{\text{causal}}\right)V \qquad M_{\text{causal}}(i,j) = \begin{cases} 0 & \text{if } j \leq i \\ -\infty & \text{if } j > i \end{cases} \tag{8.6}$$

Temporal order is encoded using **Rotary Positional Embeddings (RoPE)**, which inject relative positional information directly into the attention mechanism by rotating query and key representations:

$$Q' = R(T)Q \qquad K' = R(T)K \tag{8.7}$$

By applying RoPE to queries and keys, the model explicitly preserves the exact procedural ordering of actions, which is critical for step anticipation where the validity of each prediction depends on its precise position within the sequence. This mechanism encodes relative step distances directly within self-attention, avoiding explicit positional embeddings while ensuring strict order-awareness and consistency with procedural dynamics.

The sequence of contextualised hidden states is aggregated through a learned attention temporal pooling mechanism, which adaptively weights each past step according to its relevance for next-step prediction. Instead of relying on the final token representation, the model computes a weighted summary of the entire history:

$$\alpha_i = \text{softmax}(w(\mathbf{h}_i)), \qquad \mathbf{z} = \text{MLP}\left(\sum_i \alpha_i \mathbf{h}_i\right) \tag{8.8}$$

This produces a compact yet informative representation of the procedural context, allowing the model to focus on the most discriminative past actions.

Compared to recurrent baselines, CRAT enables direct long range interactions between distant steps through causal self-attention, while preserving strict online causality and low inference latency. Relative positional encoding further improves robustness to variable length histories, making the model a reliable and efficient task trained alternative for online procedural anticipation.

8.2.4 Causal Transformer with RoPE and Graph Bias (CRAT-GB)

Building upon the Causal RoPE Anticipation Transformer (CRAT), we extend the model with a graph-based transition prior learned from training data.
The resulting architecture, referred to as **CRAT-GB**, integrates dataset procedural statistics directly into next-step prediction while preserving full causality.
The transition prior is represented as a probability matrix

$$A \in \mathbb{R}^{V \times V} \qquad A_{i,j} \approx P(a_{t+1} = j \mid a_t = i) \tag{8.9}$$

which encodes the empirical likelihood of observing step j after step i. The matrix is computed from empirical transition counts and then row-wise normalised, ensuring scale invariance and allowing the transition prior to be consistently incorporated as a probabilistic bias.

$$A_{i,j} = \frac{N_{i \to j}}{\sum_k N_{i \to k}}, \qquad \sum_j A_{i,j} = 1 \tag{8.10}$$

After temporal modelling and attention pooling, the CRAT backbone produces next-step logits:

$$\mathbf{z} = \mathrm{MLP}(\mathrm{AttnPool}(\mathbf{h}_{1:T})) \tag{8.11}$$

The graph prior is injected at the logit level via an additive penalty term governed by a scalar β, which controls the trade-off between learned contextual representations and empirical transition probabilities.

$$\mathbf{z}' = \mathbf{z} - \beta\,(1 - \mathbf{A}[a_{\text{prev}}]) \;\Rightarrow\; \begin{cases} A_{a_{\text{prev}},j} = 1 & \Rightarrow \text{no penalty} \\ A_{a_{\text{prev}},j} \approx 0 & \Rightarrow \text{max penalty} \end{cases} \tag{8.12}$$

where a_{prev} denotes the last observed step and $A[a_{\text{prev}}]$ is the corresponding row of the transition matrix. Transitions with high empirical probability incur little or no penalty, whereas unlikely transitions are strongly suppressed.

By combining contextual temporal reasoning with dataset transition statistics, CRAT-GB enhances procedural coherence and stabilises next-step prediction, particularly under noisy or imperfect step histories. Importantly, the integration operates entirely within the causal forward pass, preserving low inference latency and full suitability for online deployment.

8.3 LLM Step Anticipation Results

This section presents the results obtained with **LLM based step anticipation** using the proposed prompting strategies on **Llama 3.2** (1B and 3B), and compares them with state-of-the-art approaches that employ LLMs as anticipation modules, such as PREGO[10] and TI-PREGO[11]. In our formulation, the LLM receives a symbolic prefix of the executed procedure and outputs a single predicted next step. A mistake is triggered whenever the recognised step differs from the anticipated continuation.

To evaluate the behaviour of LLM anticipation under different conditions, we conduct experiments using both ground-truth step sequences (*Oracle*), in order to assess intrinsic next-step prediction capability, and sequences predicted by the recognition module (*MiniRoad*[21]), reflecting the realistic online deployment scenario where anticipation operates on noisy recognised histories.

			Assembly101-O			**Epic-tent-O**		
	Step Recog.	**Step Antic.**	**Precision$_b$**	**Recall**	**F1$_b$**	**Precision$_b$**	**Recall**	**F1$_b$**
THESIS	*Oracle*	*Llama 3.2-1B with DPC*	51.55	97.79	67.51	27.2	95.3	42.32
THESIS	*Oracle*	*Llama 3.2-1B with SC*	53.10	**98.90**	69.1	28.4	96.13	43.85
THESIS	*Oracle*	*Llama 3.2-1B with 2PV*	51.69	98.34	67.76	25.61	91.17	39.99
THESIS	*Oracle*	*Llama 3.2-1B with MHC*	52.03	98.34	68.06	25.98	90.43	40.36
THESIS	*Oracle*	*Llama 3.2-3B with DPC*	55.70	93.92	69.93	28.17	96.1	43.57
THESIS	*Oracle*	*Llama 3.2-3B with SC*	54.32	98.34	69.98	30.1	96.88	45.93
THESIS	*Oracle*	*Llama 3.2-3B with 2PV*	54.52	97.24	69.87	25.01	94.67	39.57
THESIS	*Oracle*	*Llama 3.2-3B with MHC*	54.19	96.69	69.45	23.77	95.03	38.03
PREGO [10]	*Oracle*	*DeepSeek-R1* [45]	51.7	98.3	67.8	23.8	100	38.4
PREGO [10]	*Oracle*	*Llama 3.1*	56.8	89.6	69.5	28.0	92.0	42.9
TI-PREGO[11]	*Oracle*	*Llama 3.1*	**60.4**	97.8	**74.7**	**40.7**	**100**	**57.8**
THESIS	*MiniRoad*[21]	*Llama 3.2-1B with DPC*	49.86	93.37	65.0	19.88	91.03	32.63
THESIS	*MiniRoad*[21]	*Llama 3.2-1B with SC*	50.36	91.71	65.2	20.92	91.23	34.04
THESIS	*MiniRoad*[21]	*Llama 3.2-1B with 2PV*	50.24	91.71	64.91	20.00	88.57	32.63
THESIS	*MiniRoad*[21]	*Llama 3.2-1B with MHC*	50.08	91.16	64.65	20.14	92.25	33.06
THESIS	*MiniRoad*[21]	*Llama 3.2-3B with DPC*	50.03	88.4	64.12	20.18	91.33	33.06
THESIS	*MiniRoad*[21]	*Llama 3.2-3B with SC*	50.61	90.61	64.95	21.02	92.15	34.23
THESIS	*MiniRoad*[21]	*Llama 3.2-3B with 2PV*	48.92	84.53	61.97	19.45	93.03	32.17
THESIS	*MiniRoad*[21]	*Llama 3.2-3B with MHC*	49.86	90.61	64.18	20.85	90.08	33.86
PREGO [10]	*MiniRoad*[21]	*DeepSeek-R1*[45]	51.2	96.5	65.7	20.3	93.3	33.6
PREGO[10]	*MiniRoad*[21]	*Llama 3.1*	51.3	96.8	67.1	20.6	93.3	33.7
TI-PREGO[11]	*MiniRoad*[21]	*Llama 3.1*	**51.5**	**97.2**	**67.4**	**21.2**	**93.3**	**34.6**

Table 8.3.1: Comparison of LLM anticipation methods under Oracle (ground-truth) and MiniRoad (OAD) step sequences on Assembly101-O and Epic-tent-O

Table 8.3.1 shows that, under *Oracle* recognition, both Llama 3.2 variants achieve very high recall on Assembly101-O ($\approx$ 96–99%). This indicates that LLM anticipation can reliably infer the correct procedural continuation when provided with clean step histories. Among the proposed prompting strategies, **Self-Consistency via Dual Internal Experts (SC)** provides the best overall balance, achieving the highest recall with the 1B model (98.90) while maintaining competitive precision and F1 scores.

Increasing the model size from 1B to 3B mainly improves precision, resulting in F1 scores close to 70. In particular, the **3B + SC** configuration outperforms PREGO[10] with DeepSeek-R1[45] and achieves stronger results than PREGO[10] with Llama 3.1[46] on several metrics. Although TI-PREGO[11] remains the strongest approach overall, the proposed configurations significantly reduce the performance gap despite relying on smaller models.

A similar trend can be observed on Epic-tent-O when using the Oracle recognizer, although precision values are consistently lower than those observed on Assembly101-O across all methods. In this setting, the **3B + SC** configuration again provides the best results among the proposed methods ($\text{Precision}_b = 30.1$, $\text{F1}_b = 45.93$), outperforming PREGO[10] with DeepSeek-R1[45] and Llama 3.1, while TI-PREGO[11] remains the strongest approach overall.

When the anticipation module operates on step sequences predicted by the recognizer MiniRoad[21], performance decreases as expected due to recognition noise and imperfect procedural histories. Despite this, the Llama 3.2 anticipators still maintain high recall ($\approx$ 88–93% on Assembly101-O and $\approx$ 90–93% on Epic-tent-O), showing good robustness to upstream errors. Precision becomes the main limitation, particularly on Epic-tent-O, where values remain around 20–21%. Even in this more realistic setting, the **SC** prompting strategy remains the most reliable among the proposed approaches, achieving the best F1_b scores (64.95 on Assembly101-O and 34.23 on Epic-tent-O with the 3B model). These results are competitive with PREGO[10] and close to TI-PREGO[11].

Overall, the results highlight that the prompting strategies proposed in this thesis, combined with compact Llama 3.2 models (1B and 3B), achieve competitive performance compared to recent approaches such as PREGO[10] and TI-PREGO[11]. While these methods rely on substantially larger LLMs, the proposed configurations achieve competitive performance with significantly fewer parameters, and in some cases outperform existing baselines under the same evaluation protocol.

More generally, the results indicate that LLMs can effectively serve as anticipation modules when guided by well designed prompts. Even without training, they are able to capture procedural structure and predict plausible next steps, making them suitable components for online procedural mistake detection systems.

8.4 Results Step Anticipation: Trained vs Non-Trained Models

In this section, we analyse the performance of the anticipation module considering both non-trained LLM approaches and task trained models. Results are reported under both ideal conditions (correct step sequences) and realistic deployment using recognised sequences.

			Assembly101-O			Epic-tent-O		
	Step Recog.	**Step Antic.**	**Precision$_b$**	**Recall**	**F1 score$_b$**	**Precision$_b$**	**Recall**	**F1 score$_b$**
LLMs Models								
THESIS	*Oracle*	*Llama 3.2-1B with SC*	53.10	**98.90**	69.1	28.4	96.13	43.85
THESIS	*Oracle*	*Llama 3.2-3B with SC*	54.32	98.34	69.98	30.1	96.88	45.93
PREGO [10]	*Oracle*	*DeepSeek-R1*[45]	51.7	98.3	67.8	23.8	100	38.4
PREGO[10]	*Oracle*	*Llama 3.1*[46]	56.8	89.6	69.5	28.0	92.0	42.9
TI-PREGO[11]	*Oracle*	*Llama 3.1*[46]	**60.4**	97.8	**74.7**	**40.7**	**100**	**57.8**
THESIS	*OGTN*	*Llama 3.2-1B with SC*	51.02	94.72	66.32	22.25	87.83	35.51
THESIS	*OGTN*	*Llama 3.2-3B with SC*	51.21	95.81	66.75	**24.47**	91.76	**37.38**
PREGO[10]	*MiniRoad*[21]	*DeepSeek-R1*[45]	51.2	96.5	65.7	20.3	93.3	33.6
PREGO[10]	*MiniRoad*[21]	*Llama 3.1*[46]	51.3	96.8	67.1	20.6	93.3	33.7
TI-PREGO[11]	*MiniRoad*[21]	*Llama 3.1*[46]	**51.5**	**97.2**	**67.4**	21.2	**93.3**	34.6
Trained Models								
BERT[28] (fine-tuned)	*Oracle*		93.2	20.0	32.9	71.6	5.6	10.4
TGML-DO[15]	*Oracle*		**77.82**	90.4	**83.64**	**91.24**	27.3	42.03
THESIS	*Oracle*	*CRAT*	60.58	**91.93**	73.03	52.38	**73.33**	61.11
THESIS	*Oracle*	*CRAT-GB*	60.83	**91.93**	73.21	61.74	66.67	**64.11**
TGML-DO[15]	*MiniRoad*[21]		**53.77**	37.3	44.05	**96.55**	14.1	24.60
THESIS	*OGTN*	*CRAT*	52.11	**98.68**	**68.02**	39.93	46.67	43.03
THESIS	*OGTN*	*CRAT-GB*	51.87	96.69	67.52	42.45	**48.53**	**45.29**

Table 8.4.1: Comparison of trained and non-trained step anticipation models under Oracle (ground-truth) and recognised step sequences using different recognition backbones (OGTN and MiniRoad) on Assembly101-O and Epic-tent-O.

Table 8.4.1 shows that trained anticipation models generally achieve higher precision and F1 scores than non-trained LLM approaches when evaluated on ground-truth step sequences. Among the models proposed in this thesis, **CRAT-GB** obtains the best overall performance in this setting, reaching 73.21 F1$_b$ on Assembly101-O and 64.11 on Epic-tent-O, clearly outperforming the Llama 3.2 configurations evaluated under the same conditions. The graph-based transition modelling slightly improves the results compared to CRAT, particularly on Epic-tent-O, suggesting that explicitly modelling step transitions helps maintain more coherent procedural predictions.

Among the trained baselines, the fine-tuned **BERT**[28] model achieves very high precision but extremely low recall, which results in poor F1 scores and indicates limited capability to anticipate the next step reliably. In contrast, **TGML-DO**[15] obtains the best performance on Assembly101-O when ground-truth steps are used

(83.64 $F1_b$), thanks to its strong modelling of procedural transitions. Nevertheless, the proposed CRAT and CRAT-GB models remain competitive overall and show stronger performance on Epic-tent-O, achieving higher recall and F1 scores. This suggests that the proposed architectures provide a better balance between precision and recall and generalize more effectively across different procedural datasets.

Non-trained LLMs show very high recall but lower precision, especially on Epic-tent-O. Nevertheless, the best Llama 3.2 configurations remain competitive with PREGO[10], indicating that structured prompting alone can provide useful anticipation behaviour without task-specific training.

When moving from ground-truth to recognised step sequences, performance decreases for all methods due to the propagation of recognition errors to the anticipation module. In this more realistic setting, anticipation models must handle noisy and sometimes inconsistent step histories, which makes prediction significantly more challenging. Among the trained models, **CRAT** achieves the best F1 score on Assembly101-O (68.02), while **CRAT-GB** performs best on Epic-tent-O (45.29). These correspond to the highest results among the proposed anticipation solutions when using a real step recognizer, indicating that graph-based transition modelling helps maintain more coherent predictions under recognition noise.

In the Epic-tent-O experiments, the choice of step recognizer also plays an important role. The **OGTN** recognizer used in this thesis produces more stable and temporally consistent step predictions than the MiniRoad[21] baseline, reducing the noise propagated to the anticipation module and leading to more reliable step histories and improved anticipation performance.

For the LLM approaches, recall remains high even when recognised sequences are used, but precision decreases more noticeably. The best Llama 3.2 configuration (3B with SC) achieves F1 scores close to PREGO and TI-PREGO despite relying on a different recognition backbone. However, it does not surpass the trained CRAT models when recognition noise is present.

Overall, trained anticipation models provide higher precision and more stable behaviour under realistic online conditions. In particular, they show greater robustness to recognition noise and consistently achieve higher F1 scores when predicted step sequences are used. LLM-based approaches, on the other hand, still demonstrate good flexibility: even without task-specific training, compact Llama 3.2 models can obtain competitive results when guided by structured prompts. However, their performance remains generally lower than that of the trained models, indicating that task-specific learning of procedural dynamics leads to more reliable anticipation for procedural mistake detection.

In this section, we compare the proposed PMD framework with state-of-the-art approaches in terms of **model size** and **inference speed**, providing a practical perspective on deployment beyond recognition and mistake detection performance.

⋆ PMD proposed in this thesis

Model	**Recog. (M)**	**Antic. (B)**	**Total (B)**
MiniRoad[21] – PREGO[10] – LLaMA 3.1[46]	17.9	8	8.018
MiniRoad[21] – TI-PREGO[11] – LLaMA 3.1[46]	17.9	8	8.018
MiniRoad[21] – TGML-DO[15]	17.9	**0.005**	0.0229
OGTN⋆ – LLaMA 3.2 1B	**3.3**	1	1.0033
OGTN⋆ – LLaMA 3.2 3B	**3.3**	3	3.0033
OGTN⋆ – CRAT⋆	**3.3**	0.0126	**0.0159**
OGTN⋆ – CRAT-GB⋆	**3.3**	0.0126	0.0159

Table 8.4.2: Comparison of model sizes for recognition and anticipation modules

Table 8.4.2 compares the size of the recognition and anticipation modules across the considered pipelines. In LLMs solutions such as PREGO[10], TI-PREGO[11], and the Llama 3.2 configurations, most parameters are concentrated in the language model, which dominates the total model size despite the relatively lightweight MiniRoad[21] recognizer.
In contrast, the proposed framework relies on a compact recognition backbone and lightweight anticipation models. OGTN provides the smallest recognition module (3.3M parameters), significantly lighter than MiniRoad[21], while CRAT and CRAT-GB require only 12.6M parameters. As a result, the complete OGTN+CRAT pipeline remains orders of magnitude smaller than LLM approaches and lighter than TGML-DO[15] configurations on the recognition side, while maintaining competitive anticipation performance.

Table 8.4.3 reports the inference times of the evaluated PMD pipelines. TGML-DO[15] is the fastest configuration because it operates directly on the sequence of recognized steps without introducing a temporal aggregation stage. As illustrated in Fig. 8.1, step transitions are detected as soon as the recognizer changes its predicted label and verified through a lightweight graph-based precondition check, resulting in almost negligible additional latency.

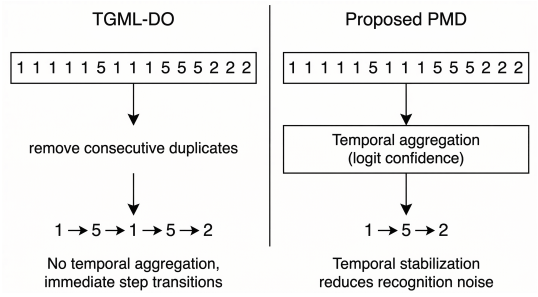


Figure 8.1: Comparison between TGML-DO and the proposed PMD framework

However, the absence of temporal aggregation in TGML-DO[15] explains part of this speed advantage. While this design performs well when reliable step sequences are available (e.g., Oracle recognition), in realistic conditions it becomes more sensitive to recognition noise, where short label fluctuations may be interpreted as spurious step transitions.
In contrast, as illustrated in Fig. 8.1, the proposed framework aggregates frame-level predictions to obtain a stable step sequence before performing procedural reasoning. Although this introduces moderate latency, it substantially improves robustness under realistic recognition noise.

PREGO[10] and TI-PREGO[11] also rely on a frame-to-step aggregation stage before anticipation can be performed. They employ the NOMA strategy, which aggregates predictions over blocks of 200 frames before committing a step.
Given the video frame rate of 30 FPS, this corresponds to an aggregation delay of approximately 6.67 s. Moreover, NOMA operates on discrete step labels rather than recognition logits, preventing early stabilization of predictions.

The proposed framework instead adopts VTP aggregation over recognition logits, enabling earlier stabilization of step predictions and reducing the delay to 3.34 s, roughly half the waiting time required by PREGO[10] and TI-PREGO[11].

Consequently, while TGML-DO[15] remains the fastest method due to the absence of aggregation, the proposed OGTN + VTP + CRAT/CRAT-GB pipelines provide a substantially faster response than prompt-based LLM approaches while producing more reliable step sequences.

⋆ PMD proposed in this thesis

Model	**Recognition(s/frame)**	+	**Aggregation(s)**	+	**Anticipator (s/step)**	=	**Total speed(s)**
MiniRoad[21] – NOMA – PREGO[10] – GPT-3.5	0.0000873	+	6.67	+	1.436	=	8.096
MiniRoad[21] – NOMA – PREGO[10] – DeepSeek-R1[45]	0.0000873	+	6.67	+	0.612	=	7.272
MiniRoad[21] – NOMA – PREGO[10] – LLaMA 3.1	0.0000873	+	6.67	+	0.216	=	6.876
MiniRoad[21] – NOMA – TI-PREGO[11] – LLaMA 3.1	0.0000873	+	6.67	+	0.315	=	6.975
MiniRoad[21] – NOMA – TGML-DO[15]	0.0000873	+	–	+	**0.00015**	=	**0.0002373**
OGTN⋆ – VTP – LLaMA 3.2 1B	**0.0000649**	+	**3.34**	+	0.11425	=	3.45
OGTN⋆ – VTP – LLaMA 3.2 3B	**0.0000649**	+	**3.34**	+	0.18931	=	3.52
OGTN⋆ – VTP – CRAT⋆	**0.0000649**	+	**3.34**	+	0.0058	=	**3.3459**
OGTN⋆ – VTP – CRAT-GB⋆	**0.0000649**	+	**3.34**	+	0.0075	=	3.34576

Table 8.4.3: Inference speed comparison between the proposed PMD pipeline and state-of-the-art approaches.

Chapter 9

Ablation

This chapter reports a series of ablation experiments aimed at understanding the impact of the main design choices introduced in the proposed framework. Instead of presenting only the final configuration, several alternative strategies were explored for the different components of the pipeline in order to evaluate their individual contribution.

The analysis focuses on four aspects of the system. First, we evaluate the use of META SAM2.1[38] as an alternative recognition module. In this setting, SAM automatically segments the objects present in the first frame and the resulting masks are then propagated across the video to obtain a form of object tracking over time.

We then analyze different architectural variants for the Online Action Detection (OAD) stage, comparing the baseline recognizer with the proposed models designed to better capture object interactions and temporal dependencies while maintaining causal inference.

Next, we study the effect of several frame-to-step aggregation strategies applied to the OTT recognizer. These experiments investigate how different temporal filtering and stabilization mechanisms influence the trade-off between recognition stability and response latency.
Finally, we explore different configurations of the anticipation module, evaluating alternative modeling choices and prompting strategies for predicting the next procedural step given the sequence of previously recognized actions.

In each experiment, the remaining parts of the pipeline are kept unchanged so that the effect of the tested component can be observed more clearly. This allows us to identify the design choices that have the largest impact on performance and the trade-offs that emerge under online and causal constraints. All experiments were conducted using NVIDIA GeForce GTX 1070 and NVIDIA T4 GPUs.

9.1 META SAM 2.1 as OAD

The **Segment Anything Model SAM2.1**[38], proposed by **Meta** is an open source model for image segmentation designed to automatically identify and isolate objects at pixel level without relying on predefined categories. Given a frame, SAM produces a set of candidate masks covering semantically meaningful regions, enabling a fully open-set and class-agnostic object representation.

In this work, SAM is used to obtain an object-centric decomposition of each frame by automatically segmenting images into candidate object masks, which are propagated across frames through temporal tracking. To remain fully agnostic to predefined categories, each segmented mask is assigned a semantic label using a Multimodal LLM, which provides a textual description of the object.
These labels are then used to train a lightweight classifier that assigns consistent object categories without requiring manual annotations.

While conceptually appealing, the SAM-based pipeline presents several practical limitations. Segmentation alone requires several seconds per frame, and the additional tracking stage further increases latency, making the approach difficult to integrate into a strictly online and low-latency setting. Moreover, egocentric videos often degrade tracking stability: motion blur, occlusions, and illumination changes may cause objects to be lost or identities to swap, leading to drifting masks and inconsistent trajectories.

Overall, although SAM provides a flexible object-centric representation, its computational overhead and tracking instability currently limit its applicability to real-time online procedural monitoring.

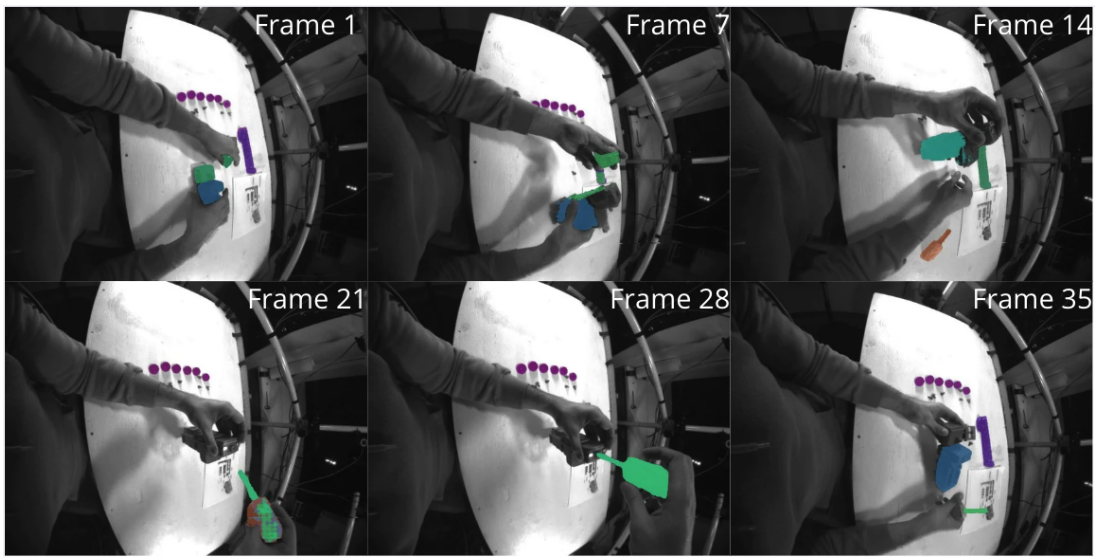


Figure 9.1: SAM mask propagation in egocentric video showing mask drift and identity swaps across frames.

9.2 Ablation OAD

9.2.1 Ablation on OGTN for OAD

This section analyzes the main design choices of the proposed OGTN architecture for Online Action Detection (OAD). Several variants were explored by progressively adjusting model capacity, graph attention configuration, regularization, and optimization parameters in order to understand which components contribute most to performance.

	Model			**Optimization**		**Detection**		**Classification**		
Run	d	depth	heads	lr	wd	mAP	mcAP	Precision	Recall	F1
v0	192	3	6	1e-4	0.05	10.96	73.08	11.47	9.17	8.82
v1	224	2	7	1e-4	0.05	11.58	74.00	11.74	10.48	10.17
v2	256	2	6	1e-4	0.02	12.28	73.96	11.89	10.93	9.99
v3	256	2	12	5e-5	0.02	12.72	74.18	**13.26**	10.75	10.06
v4	384	2	12	5e-5	0.02	11.96	74.32	12.56	10.31	9.63
v5	512	2	8	5e-5	0.02	12.26	72.62	11.65	9.72	8.61
BEST	256	2	8	5e-5	0.02	13.10	**76.01**	**13.19**	**11.17**	**10.66**

Table 9.2.1: Ablation summary for OGTN in OAD

The first configuration uses a relatively small embedding size ($d = 192$) and a deeper graph stack. This setup is stable during training but gives the lowest performance among the tested variants.

When the graph backbone is simplified and the embedding dimension is increased, the results improve. In particular, using two graph layers and $d = 256$ leads to clear gains in both detection and classification metrics.

Increasing the model size beyond this point does not help. Runs with $d = 384$ and $d = 512$ perform slightly worse, indicating that simply enlarging the representation does not improve the temporal modelling.

The number of attention heads also has some influence. Very large configurations do not show consistent improvements, while a moderate number of heads together with $d = 256$ tends to give more stable results.

Lowering the learning rate to 5×10^{-5} and using a smaller weight decay further stabilizes training and leads to the best configuration in Table 9.2.1.

Overall, the experiments suggest that a medium representation ($d = 256$) combined with a two-layer graph backbone is sufficient for this model, providing a good balance between capacity and stability.

9.2.2 Ablation on OTT for OAD

This section analyzes the main design choices of the Online Temporal Transformer (OTT) for Online Action Detection (OAD). Several configurations were explored by varying the model dimension, the number of Transformer layers, the attention heads, and the optimization parameters in order to understand which components most influence performance.

	Model				**Optimization**		**Detection**		**Classification**		
Run	d	layers	heads	$Window$	lr	wd	mAP	mcAP	Precision	Recall	F1
v0	224	8	8	128	$1e^{-4}$	0.05	9.51	73.28	9.98	9.87	8.85
v1	320	3	8	150	$1e^{-4}$	0.05	10.90	74.22	10.87	11.11	9.84
v2	320	3	8	160	$1e^{-4}$	0.02	11.75	74.75	12.45	12.82	10.81
v3	320	3	10	192	$5e^{-5}$	0.02	12.17	75.07	**12.74**	**12.81**	**11.06**
BEST	320	3	8	275	$5e^{-5}$	0.02	**13.00**	**75.12**	11.74	11.80	10.80

Table 9.2.2: Ablation summary for OTT in OAD

The initial configuration uses a relatively deep Transformer (8 layers) together with a smaller embedding dimension and a limited temporal window. Although this setup is stable during training, it produces the lowest performance among the tested variants, suggesting that the model is unnecessarily complex while still lacking sufficient temporal context.

Simplifying the architecture leads to clear improvements. Reducing the Transformer depth to three layers while increasing the embedding dimension already results in a noticeable gain across both detection and classification metrics. This indicates that, for this task, increasing the representation capacity is more beneficial than stacking additional Transformer layers.

The temporal window also plays an important role. As the window size increases, the model is able to observe a longer sequence of past frames, which progressively improves the results. This behaviour suggests that a wider temporal context helps the Transformer capture the procedural structure of the actions more effectively.

Further improvements are obtained by slightly adjusting the optimization parameters. Lowering the learning rate to 5×10^{-5} and reducing the weight decay leads to more stable convergence and contributes to the best configuration reported in Table 9.2.2.

Overall, the ablation shows that the OTT architecture benefits from a relatively compact Transformer combined with a larger temporal window. This configuration provides a better balance between model capacity, temporal context, and training stability, leading to the strongest overall performance.

9.3 Ablation on Aggregation Strategies for OTT

To better understand the effect of temporal aggregation on the Online Temporal Transformer (OTT), we evaluate several aggregation strategies under different commit latency thresholds. Table 9.3.1 reports the Levenshtein Similarity obtained on both dataset when the maximum latency allowed for committing a predicted step varies from very low (≤2s) to relaxed conditions (>6s).

	Assembly101-O							**EpicTent-O**						
Agg Method	≤2s	≤3s	≤3.5s	≤4s	≤5s	≤6s	>6s	≤2s	≤3s	≤3.5s	≤4s	≤5s	≤6s	>6s
Label Based														
NOMA	28.32	30.50	30.96	32.48	33.94	34.38	36.98	22.04	24.50	25.54	26.79	29.63	32.80	36.14
MDF+NOMA	29.23	30.80	31.36	32.79	34.16	35.63	**37.21**	24.14	25.77	25.86	27.10	29.84	32.93	38.09
HYST	29.12	29.61	29.61	29.61	29.61	29.61	27.68	**29.72**	**34.31**	**35.69**	**36.66**	**40.28**	**40.70**	**41.10**
OMA	**30.71**	31.97	32.78	33.85	34.02	35.05	36.77	24.29	29.49	32.64	34.99	35.81	36.78	37.31
Logits Based														
Soft-OMA	30.49	32.12	33.48	**34.08**	**34.78**	35.87	36.62	25.25	30.09	31.83	33.88	36.07	36.86	37.55
EMA	28.79	**32.29**	**33.64**	33.74	34.56	**36.06**	36.36	21.63	24.30	25.97	27.91	28.52	31.33	33.93
VTP	28.96	32.22	32.92	33.49	33.85	34.71	36.23	20.92	24.55	25.67	27.63	27.93	31.31	33.07

Table 9.3.1: Levenshtein Similarity for OTT across aggregation methods under different commit-latency thresholds.

Similarly to what observed for OGTN, methods operating on logits tend to perform better on Assembly101-O, especially under low latency constraints. By smoothing the probability distribution before the final decision, these strategies reduce short term fluctuations in frame predictions and enable more stable early commits.
In particular, Soft-OMA and EMA achieve the strongest results in the low latency regime on Assembly101-O, suggesting that probability smoothing helps maintain temporal coherence when predictions must be committed quickly.

Label based methods instead operate after the argmax decision and are therefore more sensitive to frame-level noise, typically requiring slightly longer observation windows to stabilize predictions. However, some variants remain competitive: OMA performs well at very low latency, while MDF+NOMA improves results when longer latency is allowed.

A different behavior is observed on EpicTent-O. As already seen with OGTN, the Hysteresis strategy proves particularly effective, consistently achieving the highest similarity across most latency thresholds thanks to its stronger commitment mechanism once a label becomes dominant.

Overall, these results confirm that aggregation plays an important role in balancing stability and latency: logits strategies are generally preferable when early decisions are required, while stronger commitment mechanisms can be advantageous when temporal consistency becomes more important.

9.4 Ablation Step Anticipation

9.4.1 LLM Prompt Ablation

Figure 9.2 shows the impact of temperature and top-p on the $F1_b$ score for Llama 3.2 3B across the proposed prompting strategies.

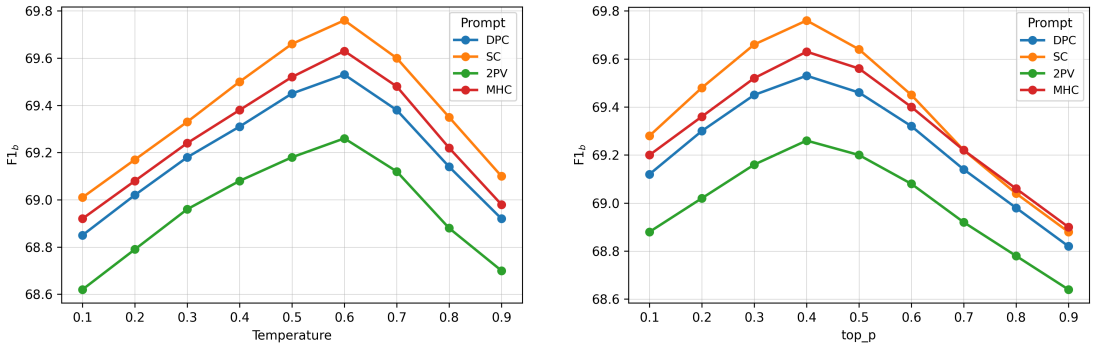


Figure 9.2: Effect of temperature (left) and top-p (right) on $F1_b$ for Llama 3.2 3B with different prompting strategies (DPC, SC, 2PV, MHC).

Four prompting strategies are evaluated. Direct Procedural Completion (DPC) predicts the next step directly from the context and serves as a baseline. Self-Consistency (SC) generates multiple candidates and selects the most consistent one, while Two-Pass Verification (2PV) first produces a prediction and then refines it in a second pass. Multi-Hypothesis Coherence (MHC) instead evaluates several possible continuations and selects the most coherent one.

When varying the temperature, all strategies follow a similar trend. Performance increases from 0.1 up to around 0.6, after which it slightly decreases. Very low temperatures make the model overly deterministic, while moderate stochasticity helps explore alternative procedural continuations. SC consistently achieves the best results and reaches the highest score at $T = 0.6$ ($F1_b = 69.76$). DPC and MHC show very similar behaviour and remain close to SC, while 2PV stays slightly below the other strategies across the entire range.

A similar pattern emerges when varying the top-p parameter. The best performance is obtained around $top - p = 0.4$: lower values restrict the sampling space too much, while larger values introduce additional variability and slightly degrade the predictions. Also in this case, SC provides the best results, followed closely by MHC and DPC, whereas 2PV remains the weakest configuration.

Overall, the results indicate that moderate stochasticity leads to the most reliable predictions. Across both parameters, SC consistently performs best, although the gap with DPC and MHC remains relatively small.

9.4.2 Ablation Trained Anticipation Module

This section analyzes the behaviour of the trained Transformer-based anticipation models. We compare CausalRoPETransformer (CRAT) and its GraphBias variant (CRAT-GB), which introduces a structural bias to capture dependencies between procedural steps. Table 9.4.1 reports the results obtained under different architectural and training configurations. All experiments are evaluated using OAD ground truth step sequences, allowing us to analyze the anticipation models without the noise introduced by recognition errors.

	Model			**Training**		**Classification**			
Run	d	layers	heads	batch	lr	Acc	Precision$_b$	Recall	F1$_b$
CRAT									
v0	256	2	8	16	$3e^{-4}$	44.77	51.44	**98.68**	67.93
v1	384	2	6	16	$1e^{-4}$	41.92	51.54	96.68	67.01
v2	384	2	8	16	$3e^{-4}$	45.67	52.82	95.70	68.23
v3	448	2	8	16	$3e^{-4}$	46.03	53.31	95.01	68.54
v4	448	4	8	32	$1e^{-4}$	51.73	55.39	96.32	69.13
v5	448	5	8	48	$3e^{-4}$	55.73	56.26	93.72	70.54
BEST	448	5	8	16	$3e^{-4}$	**57.47**	**60.58**	91.93	**73.03**
CRAT-GB									
v0	256	2	8	16	$3e^{-4}$	46.53	53.44	96.13	66.93
v1	384	2	6	16	$1e^{-4}$	48.92	53.82	**96.71**	67.21
v2	384	2	8	16	$3e^{-4}$	49.03	53.21	95.23	67.13
v3	448	2	8	16	$3e^{-4}$	50.03	54.44	94.97	68.54
v4	448	4	8	32	$1e^{-4}$	53.73	56.89	95.89	70.43
v5	448	5	8	48	$3e^{-4}$	56.73	56.26	94.72	71.54
BEST	448	5	8	16	$3e^{-4}$	**57.47**	**60.83**	91.93	**73.21**

Table 9.4.1: Ablation comparison between CausalRoPETransformer (CRAT) and its GraphBias variant (CRAT-GB) across different training configurations.

For the CRAT model, increasing the model capacity generally improves performance. Moving from smaller configurations ($d = 256$) to larger representations ($d = 448$) leads to higher accuracy and F1 scores, while deeper models also provide better temporal modelling of the step sequence.

The best CRAT configuration ($d = 448$, 5 layers, 8 heads) reaches an F1 score of 73.03 with an accuracy of 57.47. Across all runs the recall remains consistently

high, indicating that the model captures the overall progression of procedural steps, while most improvements come from higher precision.

The GraphBias variant follows a similar trend but achieves slightly better results. The best CRAT-GB configuration reaches an F1 score of 73.21 with a precision of 60.83 while maintaining the same recall as the base model, suggesting that incorporating structural relations between steps helps refine the predictions.

Overall, increasing the Transformer capacity improves anticipation performance, while the addition of a lightweight graph bias provides a small but consistent gain.

Chapter 10

Conclusion and Future Work

10.1 Conclusion

This work addressed online Procedural Mistake Detection (PMD) in egocentric video under realistic deployment constraints, treating the problem as an integrated system where recognition, temporal aggregation, and anticipation interact to determine mistake detection performance in streaming settings.

On the recognition side, the proposed OGTN architecture improves robustness to egocentric perceptual noise by combining object-centric representations with lightweight temporal modelling under strict causal constraints, producing more stable step predictions and reducing error propagation to downstream modules.

The experimental analysis confirms the structural trade-off between temporal context and latency: increasing context improves stability but introduces delay. Rather than accepting this as fixed, the thesis quantifies and optimises this balance, showing that aggregation latency can be reduced to $\leq$4 seconds while preserving structural consistency comparable to slower baselines.

For anticipation, the comparison between non-trained LLMs and task-trained predictors reveals complementary strengths. LLMs provide flexible next-step reasoning without fine-tuning, while task-trained models show more stable behaviour under recognition noise.

Finally, the proposed design demonstrates that effective PMD systems can be implemented with limited computational resources. The complete framework remains lightweight (below 20M parameters), enabling low-latency operation and deployment on devices with constrained resources.

Overall, the thesis highlights that online PMD should be evaluated not only through classification metrics, but also in terms of temporal coherence, latency, and deployability.

10.2 Limitations

Despite the encouraging results, several limitations remain.

First, the adopted One-Class formulation assumes that correct procedural executions are sufficiently representative of real-world variability. In practice, procedural style may vary across users, environments, and skill levels, potentially affecting both recognition stability and anticipation accuracy. Handling such variability without explicit supervision remains an open challenge.

Second, although the anticipation models use the full observed history, their predictions are still based only on learned sequence patterns. They do not explicitly model task structure, safety rules, or the underlying goals of the procedure.
While this works well in structured assembly benchmarks, more complex real-world scenarios may require a deeper understanding of task intent, contextual constraints, and longer-term dependencies.

Third, mistake detection is formulated as a binary inconsistency between recognised and anticipated steps. This decision rule does not account for uncertainty levels, mistake severity, or partial deviations, which may be relevant in real assistive systems.

Finally, despite the encouraging quantitative results, the achieved performance is not yet sufficient to enable fully autonomous assistance. Recognition noise, residual anticipation errors, and binary mistake signalling may still lead to incorrect or unstable interventions. For real assistive deployment, higher reliability, calibrated uncertainty estimation, and more robust error handling would be required.

10.3 Future Work

Although the proposed framework shows promising results, several aspects could be further explored in future work.

One possible extension concerns the interaction between the different modules of the system. In the current pipeline, recognition, aggregation, and anticipation are trained independently and only connected during inference. While this modular design simplifies experimentation, it may limit the ability of the system to learn representations optimized for the final mistake detection task. Future work could therefore explore joint or end-to-end training strategies, allowing anticipation signals to influence earlier stages of the pipeline.

Another aspect concerns the role of uncertainty in mistake detection. The current approach mainly relies on discrete step predictions and mismatches between predicted and expected actions. However, visual predictions are often uncertain, especially in egocentric videos affected by motion blur, occlusions, or rapid viewpoint changes. Incorporating confidence estimation or probabilistic reasoning could allow the system to produce more informative alerts instead of relying solely on binary decisions.

Another limitation of the current pipeline concerns the robustness of step recognition in challenging egocentric conditions. Motion blur, occlusions, and fast camera movements often introduce noise in frame-level predictions that propagates through the system. Exploring stronger temporal regularization strategies or more stable object-centric representations could help improve recognition stability.

Finally, the evaluation in this thesis focuses on two procedural benchmarks. Extending the analysis to additional datasets would help better assess how the proposed approach generalizes to different tasks and environments, providing further insight into the robustness of the framework.

10.4 Closing Remarks

Online procedural mistake detection sits at the intersection of perception, temporal modeling, and symbolic reasoning. Designing systems that operate causally, efficiently, and robustly in egocentric settings requires careful coordination across these dimensions.

This thesis demonstrates that such coordination is achievable. By explicitly addressing latency, stability, and procedural coherence as first-class design objectives, it contributes toward scalable and realistic intelligent assistance systems capable of monitoring and supporting structured human activities in real time.

At the same time, online PMD remains an open research problem. Variability in real-world procedures, perceptual uncertainty, and the need for reliable autonomous intervention continue to pose significant challenges. The results presented here should therefore be viewed as a step toward robust online deployment, rather than a definitive solution.

Bibliography

[1] Reza Ghoddoosian, Hongyang Li, Mohit Bansal, Zhe Wang, and Jiaying Liu. «ATA: Activity Task Analysis for Offline Procedural Mistake Detection». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023 (cit. on p. 7).

[2] 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». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022 (cit. on pp. 7, 8, 12–14).

[3] Youngkyoon Jang, Brian Sullivan, Casimir Ludwig, Iain Gilchrist, Dima Damen, and Walterio Mayol-Cuevas. «EPIC-Tent: An Egocentric Video Dataset for Camping Tent Assembly». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. 2019 (cit. on pp. 7, 8, 12, 14).

[4] Xin Wang et al. «HoloAssist: an Egocentric Human Interaction Dataset for Interactive AI Assistants in the Real World». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023 (cit. on p. 7).

[5] 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*. 2024, pp. 18655–18666 (cit. on p. 8).

[6] Konstantinos Bacharidis and Antonis A Argyros. «Vision-Based Mistake Analysis in Procedural Activities: A Review of Advances and Challenges». In: *arXiv preprint arXiv:2510.19292* (2025) (cit. on p. 8).

[7] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. «Deep One-Class Classification». In: *Proceedings of the 35th International Conference on Machine Learning*. Proceedings of Machine Learning Research. PMLR, 2018 (cit. on p. 8).

[8] Mahmudul Hasan, Jonghyun Choi, Jan Neumann, Amit K. Roy-Chowdhury, and Larry S. Davis. «Learning Temporal Regularity in Video Sequences». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016 (cit. on p. 8).

[9] Waqas Sultani, Chen Chen, and Mubarak Shah. «Real-World Anomaly Detection in Surveillance Videos». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018 (cit. on p. 8).

[10] Alessandro Flaborea, Guido Maria D'Amely di Melendugno, Leonardo Plini, Luca Scofano, Edoardo De Matteis, Antonino Furnari, Giovanni Maria Farinella, and Fabio Galasso. «PREGO: Online Mistake Detection in PRocedural EGOcentric Videos». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2024, pp. 18483–18492 (cit. on pp. 8, 11, 12, 39, 45–50).

[11] Leonardo Plini, Luca Scofano, Edoardo De Matteis, Guido Maria D'Amely di Melendugno, Alessandro Flaborea, Andrea Sanchietti, Giovanni Maria Farinella, Fabio Galasso, and Antonino Furnari. «TI-PREGO: Chain of Thought and In-Context Learning for online mistake detection in PRocedural EGOcentric videos». In: *Computer Vision and Image Understanding* (2026) (cit. on pp. 8, 11, 20, 39, 45–47, 49, 50).

[12] Sijie Yan, Yuanjun Xiong, and Dahua Lin. «Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition». In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018 (cit. on pp. 8, 10).

[13] Fida Mohammad Thoker and Juergen Gall. «Skeleton Graph Contrastive Learning for Action Recognition». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021 (cit. on pp. 8, 10).

[14] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. «A Simple Framework for Contrastive Learning of Visual Representations». In: *Proceedings of the International Conference on Machine Learning (ICML)*. 2020 (cit. on pp. 8, 10).

[15] Luigi Seminara, Giovanni Maria Farinella, and Antonino Furnari. «Differentiable Task Graph Learning: Procedural Activity Representation and Online Mistake Detection from Egocentric Videos». In: *The Thirty-eighth Annual Conference on Neural Information Processing Systems*. 2024 (cit. on pp. 8, 11, 47, 49, 50).

[16] Robin M Schmidt. «Recurrent neural networks (rnns): A gentle introduction and overview». In: *arXiv preprint arXiv:1912.05911* (2019) (cit. on p. 9).

[17] Sepp Hochreiter and Jürgen Schmidhuber. «Long Short-Term Memory». In: *Neural Computation* (1997) (cit. on p. 9).

[18] Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. «Learning phrase representations using RNN encoder–decoder for statistical machine translation». In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1724–1734 (cit. on p. 9).

[19] Yanghao Li, Cuiling Lan, Junliang Xing, Wenjun Zeng, Chunfeng Yuan, and Jiaying Liu. «Online human action detection using joint classification-regression recurrent neural networks». In: *European conference on computer vision*. 2016 (cit. on p. 9).

[20] Mingze Xu, Mingfei Gao, Yi-Ting Chen, Larry S Davis, and David J Crandall. «Temporal recurrent networks for online action detection». In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019 (cit. on p. 9).

[21] Joungbin An, Hyolim Kang, Su Ho Han, Ming-Hsuan Yang, and Seon Joo Kim. «MiniROAD: Minimal RNN Framework for Online Action Detection». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023 (cit. on pp. 9, 22–24, 30, 37, 45–50).

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. «Attention Is All You Need». In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017 (cit. on pp. 9, 10).

[23] Xiang Wang, Shiwei Zhang, Zhiwu Qing, Yuanjie Shao, Zhengrong Zuo, Changxin Gao, and Nong Sang. «Oadtr: Online action detection with transformers». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021 (cit. on p. 9).

[24] Ruixin Li, Longchuan Yan, Yuanlong Peng, and Laiyun Qing. «Lighter transformer for online action detection». In: *Proceedings of the 2023 6th International Conference on Image and Graphics Processing*. 2023 (cit. on p. 9).

[25] Zhanzhong Pang, Fadime Sener, and Angela Yao. «Context-enhanced memory-refined transformer for online action detection». In: *Proceedings of the Computer Vision and Pattern Recognition Conference*. 2025 (cit. on p. 9).

[26] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. «Object-centric learning with slot attention». In: *Advances in neural information processing systems* (2020) (cit. on p. 10).

[27] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. «Perceiver: General perception with iterative attention». In: *International conference on machine learning*. 2021 (cit. on p. 10).

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. «Bert: Pre-training of deep bidirectional transformers for language understanding». In: *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*. 2019 (cit. on pp. 10, 47).

[29] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. «Roformer: Enhanced transformer with rotary position embedding». In: *Neurocomputing* (2024) (cit. on p. 10).

[30] Zijia Lu and Ehsan Elhamifar. «Set-supervised action learning in procedural task videos via pairwise order consistency». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 19903–19913 (cit. on p. 11).

[31] Jingwei Ji, Ranjay Krishna, Li Fei-Fei, and Juan Carlos Niebles. «Action genome: Actions as compositions of spatio-temporal scene graphs». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020 (cit. on p. 11).

[32] Ivan Rodin, Antonino Furnari, Kyle Min, Subarna Tripathi, and Giovanni Maria Farinella. «Action scene graphs for long-form understanding of egocentric videos». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024 (cit. on p. 11).

[33] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. «Language Models are Few-Shot Learners». In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2020 (cit. on p. 11).

[34] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. «Chain-of-Thought Prompting Elicits Reasoning in Large Language Models». In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2022 (cit. on p. 11).

[35] Gemini Team. «Gemini 1.5: Unlocking Multimodal Understanding across Millions of Tokens». In: *arXiv preprint arXiv:2403.05530* (2024) (cit. on p. 21).

[36] An Yang et al. «Qwen2.5 Technical Report». In: *arXiv preprint arXiv:2412.15115* (2025) (cit. on pp. 21, 23).

[37] Hao Chen, Shengju Fang, Haoxuan Zhang, Ziwei Wang, Jinpeng Chen, Yixuan Huang, Tong Zhang, et al. «Expanding Performance Boundaries of Open-Source Multimodal Large Language Models with InternVL 2.5». In: *arXiv preprint arXiv:2412.05271* (2025) (cit. on pp. 21, 23).

[38] Nikhila Ravi, Sunil Gadre, Abishek Paulukaitis, Surya Bhatt, Eric Monari, Alexander Kirillov, Alexander C Berg, and Piotr Dollár. «SAM 2: Segment Anything in Images and Videos». In: *arXiv preprint arXiv:2408.00714* (2024) (cit. on pp. 21, 51, 52).

[39] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. «Deep residual learning for image recognition». In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016 (cit. on p. 21).

[40] Sergey Ioffe and Christian Szegedy. «Batch normalization: Accelerating deep network training by reducing internal covariate shift». In: *International conference on machine learning*. pmlr. 2015 (cit. on p. 21).

[41] Gemma Team, Google DeepMind. «Gemma 3 Technical Report». In: *arXiv preprint arXiv:2503.19786* (2025) (cit. on p. 23).

[42] Tianyu Yu et al. «Minicpm-v 4.5: Cooking efficient mllms via architecture, data, and training recipe». In: *arXiv preprint arXiv:2509.18154* (2025) (cit. on p. 23).

[43] Google DeepMind. *Gemini (Versione 2.0 Flash)*. 2025 (cit. on pp. 23, 24, 30).

[44] Haiwan Wei, Yitian Yuan, Xiaohan Lan, Wei Ke, and Lin Ma. «Instruction-bench: An instructional video understanding benchmark». In: *arXiv preprint arXiv:2504.05040* (2025) (cit. on pp. 24, 30).

[45] DeepSeek AI Team. «DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning». In: *arXiv preprint arXiv:2501.12948* (2025) (cit. on pp. 45–47, 50).

[46] Meta. «The Llama 3 Herd of Models». In: *arXiv preprint arXiv:2407.21783* (2024) (cit. on pp. 46, 47, 49).