CaptainCook4D: A Dataset for Understanding Errors in Procedural Activities [Supplementary Material]

Rohith Peddi [+] [*] **Shivvrat Arya** [+] **Bharath Challa** [+] **Likhitha Pallapothula** [+]

Akshay Vyas [+] **Bhavya Gouripeddi** [+] **Qifan Zhang** [+] **Jikai Wang** [+]

Vasundhara Komaragiri [+] **Eric Ragan** [×] **Nicholas Ruozzi** [+] **Yu Xiang** [+] **Vibhav Gogate** [+]

Appendices

[*] Corresponding Author , [+] = UT Dallas, [×] = University of Florida

Submitted to the 38th Conference on Neural Information Processing Systems (NeurIPS 2024) Track on Datasets and Benchmarks. Do not distribute.

A Overview

A.1 Motivation

Procedural Datasets. We present our motivation to collect a new dataset with errors. Current datasets that study procedural tasks, such as GTEA [20], Breakfast [37], CMU-MMAC [13], 50Salads [72], COIN [74], CrossTask [86], ProceL [17], EgoProceL[4], Assembly101 [18], and HowTo100M [46], encompass temporal variation in the order of the steps performed. However, these datasets are predominantly sourced from crowd-sourced online platforms, resulting in the videos often containing drastically different steps, with alterations impacting more than 30% of the content. Our interest lies in understanding errors induced by deviating from the given instruction set. To this end, we require two types of videos: normal ones that closely follow the instructions and error videos that depict deviations. Moreover, we aim to capture these videos from an ego-centric perspective to minimize the occlusions typical in third-person videos. We are primarily interested in understanding errors when the objects under the interaction continuously change shape and colour during a procedural activity.

Recent Progress. Error recognition in procedural activities has received significant traction, leading to the proposal of new datasets with errors [76, 22, 60, 63]. Although they aim to identify errors in procedural activities, they focus on tasks related to assembly and disassembly. **The activities involve objects with constant shapes and colors, which lack the desired characteristics**. The absence of such specific video resources led us to curate a dataset (Fig. 1) embodying all our desired characteristics. By focusing on cooking activities with desired characteristics, our dataset can be used to develop easily transferable algorithms for other sensitive domains, such as medicine and chemistry.

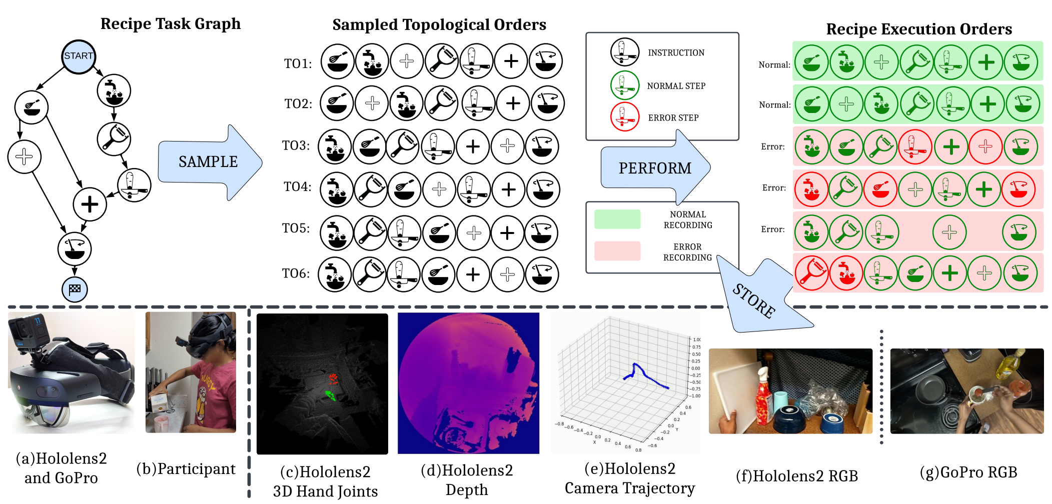

Figure 1: **Overview.** Top: We constructed task graphs for the selected recipes. These graphs facilitated sampling partial orders (cooking steps) that participants followed to perform. During the execution of some of these steps, participants induced errors that are both **intentional** and **unintentional** in nature. Bottom: On the left, we present the sensors employed for data collection, and on the right, we describe the details of the modalities of the data collected while the participant performs the recipe.

In the following sections, we will start by explaining how to use the data, including details about the structured splits of the dataset. We will then discuss comprehensive results and various analyses of the tasks we have benchmarked. Lastly, we will describe our data collection and annotation processes involving three stages: (a) Data collection planning, (b) Data collection, and (c) Data processing.

A.2 Extended Related Work

Recently, [64] proposed a method for supervised procedure learning on the **proposed dataset**. A complete survey of all the relevant tasks is outside the scope of the paper; thus, we provide a brief review of procedure understanding tasks that are of particular interest to the proposed dataset such as Error Recognition[70, 9], Multi-Step Localization and Self-Supervised Procedure Learning [15, 16, 39], Video Summarization [48], Temporal Action Segmentation [19, 40, 1, 10, 21], Object State Change Detection [71], Action Localization [87, 68], Adverb Recognition [11, 11], Task Verification (Sequence Verification [80]) [60], Long Video Understanding [30, 28, 83], Key-Step Localization [51, 44, 26], Procedure Planning [27] (Goal-Step Inference [82, 38, 55]), Self-supervised procedural knowledge extraction [49] (Visual Transformation Telling [78]), Sequence-to-sequence alignment [50, 23, 5, 7, 75, 33, 34, 62, 32, 73, 81, 69, 77] (Audio, Video synchronization [47, 8, 57, 67]), Scene Graph Anticipation [54], Temporal Adaptation, Semantic Role Labelling [3, 6, 29], Procedure Learning [85, 35, 43, 84], Action Anticipation in Procedural Videos [65, 56, 2].

A.3 Data Splits

We created diverse splits for training models on the proposed dataset where each split is based on a specific criteria ensuring diversity in the train, validation and test subsets according to it. This approach enables models to concentrate on different facets of the data. The splits are categorized as follows: (1) Recording Environment ($\mathcal{E}$), (2) Recording Person ($\mathcal{P}$), (3) Recipes ($\mathcal{R}_e$), (4) Recordings ($\mathcal{R}$), (5) Steps ($\mathcal{S}$), and (6) Recording Type ($\mathcal{R}_t$).

(1) Environment ($\mathcal{E}$). Our dataset comprises data collected from ten different environments, with a larger proportion of recordings sourced from five of these environments. We used this information to strategically divide the dataset. Recordings from these five environments were included in both the training and validation sets, while recordings from the remaining environments were allocated to the test set. We ensured a consistent balance of normal and error recordings across all three sets.

(2) Persons ($\mathcal{P}$). Eight participants compiled our dataset, each recording an equal number of videos. To facilitate a balanced distribution, we designed a split that includes recordings from two participants—who performed all the recipes—in the test set. The recordings from the remaining participants were divided between the training and validation sets.

(3) Recipes ($\mathcal{R}_e$). We meticulously divided 24 selected recipes into training, validation, and test sets based on the specific skills required for each recipe. By identifying all the essential skills needed to execute these recipes, we ensured that each set included recipes that necessitate applying these skills. This strategic division facilitates learning tasks that involve skill transfer.

(4) Recordings ($\mathcal{R}$). We categorize all recordings of a recipe into training, validation, and test sets according to a specified ratio. This split is generated randomly and varies with each iteration.

(5) Steps ($\mathcal{S}$). We compile a comprehensive dataset consisting of video segments that correspond to the steps of all recordings. This dataset is then divided into training, validation, and test splits, ensuring that steps from each recording are represented across all three splits.

(6) Recording Type ($\mathcal{R}_t$). Tasks that require a semantic understanding of errors employ methods that differentiate between normal and error recordings. The models can be trained using only normal recordings to learn the baseline behaviour and then applied to recognize errors in recordings.

B Benchmarking

We present comprehensive evaluation results and analyses for our proposed dataset on several tasks:

1. **Error Recognition**: This includes evaluations under two different settings:
   - **Zero-Shot**:
     - Error Recognition
     - Error Category Recognition
     - Anomaly Detection
   - **Supervised**:
     - Error Recognition
     - Early Error Recognition
2. **Multi-Step Localization**
3. **Procedure Learning**

B.1 Zero-Shot Error & Error Category Recognition

Error Recognition demands an accurate interpretation of actions and their consequences. This entails developing models that can semantically understand the progression of events during an activity and also assess the quality of the actions observed. Recently, Vision-Language Models (VLMs) have shown great promise in visual reasoning by combining effective visual analysis with the strong common-sense reasoning abilities of Large Language Models (LLMs). Therefore, our goal is to test the ability of recently proposed VLMs to recognize errors in video recordings of procedural activities.

Leveraging the prompt-and-predict paradigm we proposed two variants[2] $\{\mathcal{V}_1, \mathcal{V}_2\}$ for error recognition. We set up the Zero-Shot Error Recognition task as a Video Question Answering (VQA) problem. Our proposed variants $\{\mathcal{V}_1, \mathcal{V}_2\}$ primarily focused on the generation of task-specific questions, while relying on the existing state-of-the-art pre-trained VLMs for visual interpretation and the reasoning ability required (specific to visual interpretation) to answer the generated task-specific questions.

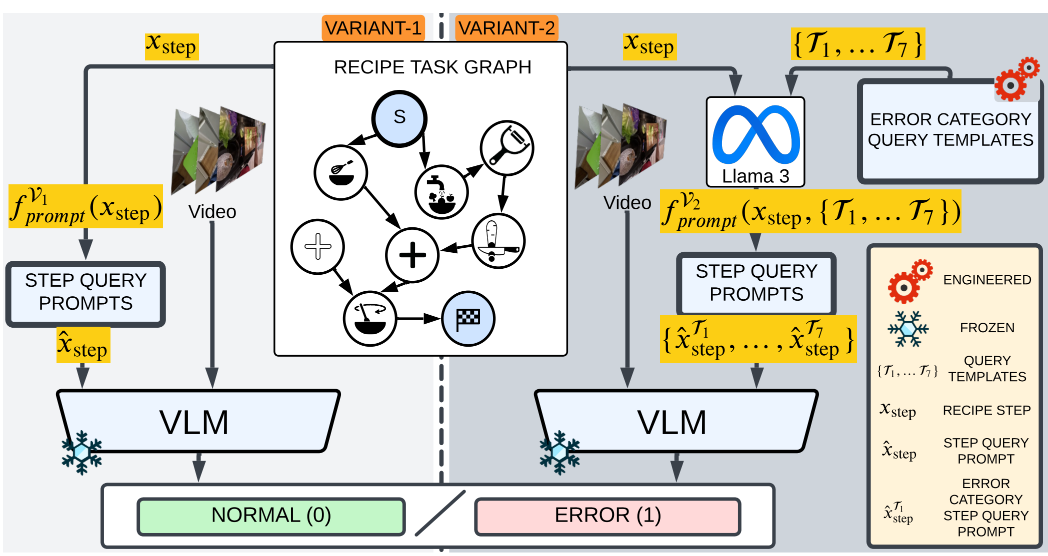

Figure 2: **ZeroShotER** evaluation pipeline of VLMs

Our proposed variants are distinguished by their use of single-prompt and multi-prompt approaches, as defined in the literature on prompt engineering. Specifically, in $\mathcal{V}_1$, we leverage the task graphs

[2]We only probe the textual inputs, leaving the visual interpretation aspects of the VLMs unchanged.

and error descriptions provided as part of annotations to construct questions (a single question prompt specific to each step of the recipe) that enquire about the completion of a recipe step in the procedural activity videos. In $\mathcal{V}_2$, instead of a single-prompt (more general questions), we adopt a prompt-ensembling strategy (more specific questions) to recognize errors that occur in recordings.

Our $\mathcal{V}_2$ can be understood as follows: Error Recognition as a task requires identification of all errors that occur in procedural activity videos. Since the space of the possible errors that can potentially occur for each procedural activity is very large and combinatorial in nature, tasking a VLM to enquire about all possible errors through more general questions leads to significantly low performance (can be observed through numbers of $\mathcal{V}_1$ in Table 1). To address this, as illustrated in Figure 2, we leveraged the structured knowledge about the categories of errors that can occur while executing a procedural activity and crafted targeted question prompts. This strategy not only guides VLMs to answer more specific questions, thereby improving the performance scores (refer Table 1) but also aids in developing a systematic framework for building error recognition models using VLMs.

VLM	**Variant**	**Acc**	**P**	**R**	**F1**
Video-LLaVa [41]	$\mathcal{V}_1$	64.3	34.2	3.9	6.7
	$\mathcal{V}_2$	52.85	36.3	49.3	41.8
TimeChat [59]	$\mathcal{V}_1$	65.0	51.11	1.15	2.26
	$\mathcal{V}_2$	43.5	34.38	69.7	46.1

Table 1: **ZeroShotER** evaluation results.

In subsequent sections, we detail the two proposed variants, $\{\mathcal{V}_1, \mathcal{V}_2\}$, including examples of the crafted question prompts for $\{\mathcal{V}_1, \mathcal{V}_2\}$. We also present our evaluation results (using standard binary classification metrics) for the tasks of Error Recognition and Error Category Recognition. We note that although we present evaluation results for two open-source VLMs, Video-LLaVa and TimeChat, our framework can be easily extended to closed-source VLMs such as GPT-4V and GeminiPro.

B.1.1 Variant-1 ($\mathcal{V}_1$):

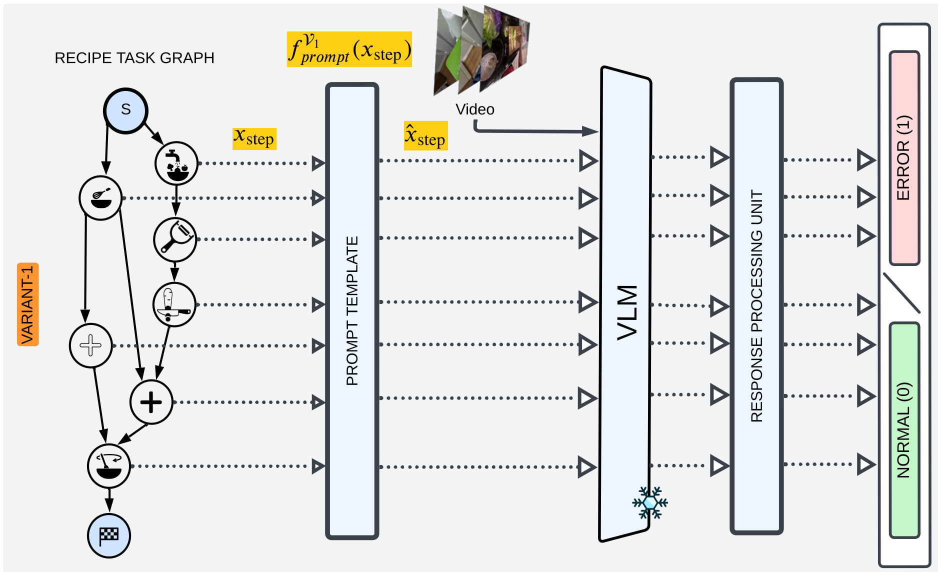

Figure 3: **ZeroShotERV1.** We outline the methodology for the proposed variant $\mathcal{V}_1$ as follows: Initially, we integrated the raw textual descriptions of steps extracted from task graphs into an engineered prompt template to create a single-question prompt specific to each step of the recipe. This prompt, along with the corresponding video, is inputted into a VLM. We analyze the responses generated from the VLM to obtain its predictions corresponding to each step of the recipe.

Task. In Figure 3, we present overview of the proposed method. The two important components of $\mathcal{V}_1$ include (a) An engineered prompt template that is specific to the chosen VLM and (b) A response processing unit. For engineering the prompt template, we employed the following methodology:

- Inspired by the *Chain of Thought* prompting strategy, We formulated a list of templates that can potentially be used for recognizing errors in procedural activity recordings.
- We selected a diverse set of videos representing every recipe type included in the dataset.
- We executed our proposed pipeline with all the prompt templates on a selected set of videos and chose the best-performing prompt template corresponding to each VLM.

We noticed that although we craft the template to generate the response in a specific format, the response generated by VLM often follows a different format; thus, we included a response processing unit to convert the generated response into a preferred format. We presented results in Table 1.

In the subsequent sections, we present the engineered templates (tailored to the specific choice of VLMs) used to construct the question prompts for each step followed by a few examples of the steps sampled from the recipes included in the proposed dataset. We built our candidates for the templates utilizing the examples provided by the authors of employed VLMs, Video-LLaVa and TimeChat.

Prompt Template. Following are the engineered prompt templates corresponding to the VLMs

- **Video-LLaVa:** ASSISTANT: {Did, Is, Does, ...} the person {perform, execute, doing, ...} the step recipe step from the recipe recipe ?
- **TimeChat:** You are given a cooking video. Please watch the video and answer the following question: {Did, Is} the person {perform, doing} the step recipe step ? Return the answer in the format of Yes or No.

Examples:

Recipe:	Cucumber Raita
Step:	*In a mixing bowl, whisk 1 cup of chilled curd until smooth. Use fresh homemade or packaged curd*
VLM:	Video-LLaVa
Prompt:	Did the person whisk 1 cup of chilled fresh homemade or packaged curd until smooth while performing the Cucumber Raita recipe?
VLM:	TimeChat
Prompt:	You are given a cooking video. Please watch the video and answer the following question: Did the person whisk 1 cup of chilled fresh homemade or packaged curd until smooth? Return the answer in the format of Yes or No.

Recipe:	Spiced Hot Chocolate
Step:	*Add 2 pieces of chocolate to the mug*
VLM:	Video-LLaVa
Prompt:	Does the person add 2 pieces of chocolate to the mug while performing Spiced Hot Chocolate recipe?
VLM:	TimeChat
Prompt:	You are given a cooking video. Please watch the video and answer the following question: Does the person add 2 pieces of chocolate to the mug? Return the answer in the format of Yes or No.

Recipe:	Tomato Mozzarella Salad
Step:	*Rinse a tomato*
VLM:	Video-LLaVa
Prompt:	Did the person rinse one tomato?
VLM:	TimeChat
Prompt:	You are given a cooking video. Please watch the video and answer the following question: Did the person rinse one tomato? Return the answer in the format of Yes or No.

B.1.2 Variant-2:

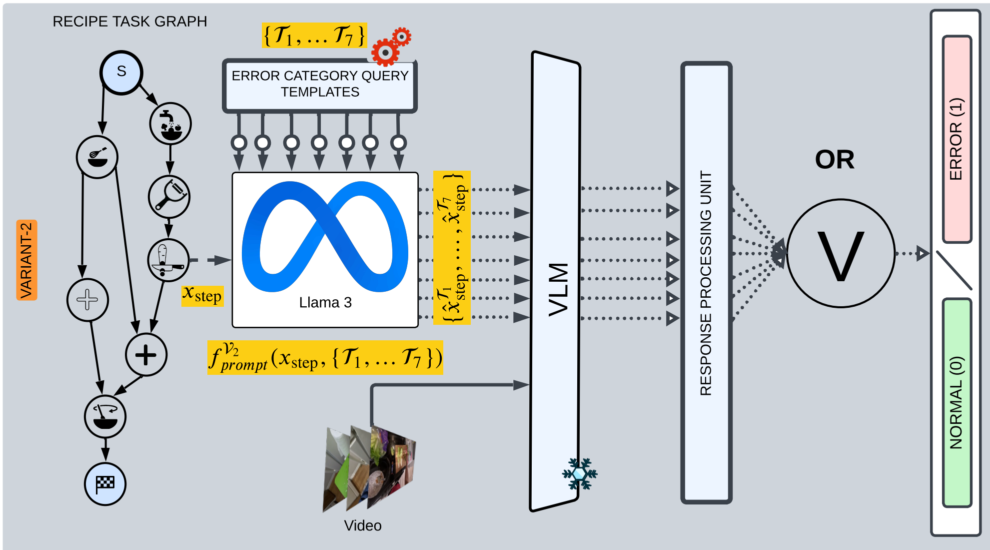

Figure 4: **ZeroShotERV2.** We outline the methodology for the proposed variant $\mathcal{V}_2$ as follows: Instead of a single-question prompt for each recipe step, we create seven question prompts, each tailored to a specific error category. Specifically, we engineered seven query templates, each corresponding to an error category. We fed these templates along with the description of each recipe step to Llama3 to generate seven tailored query prompts. We pass these error category-specific query prompts along with the videos to VLMs. We process the response generated by VLMs and apply an **OR** operation on processed responses of question prompts to obtain the final prediction for each step.

Task. In Figure 4, we present overview of the proposed variant $\mathcal{V}_2$. Important components of $\mathcal{V}_2$ are (a) Engineered error-category-specific query prompt templates tailored to VLM and (b) Final prediction generation. For engineering error-category-specific query prompt templates we followed:

- Inspired by the *Chain of Thought* prompting strategy, We formulated a list of templates that can be used for recognizing errors corresponding to *each error category* in recordings.
- We selected a diverse set of videos representing every recipe type included in the dataset.
- We executed our proposed pipeline using all error category-specific query prompt templates on a chosen set of videos. From these, we identified and selected the best-performing error category-specific query prompts for each VLM. We presented evaluation results in Table 1.
- We note that as an intermediate step, we also solve the Error Category Recognition task.

Table 2: **Error Category Recognition** evaluation results.

Error Category	**VLM**	**P**	**R**	**F1**	**Acc**
Order Error	Video-LLaVA	16.3	35.3	22.3	65.5
	TimeChat	17.2	6.25	9.2	82.6
Preparation Error	Video-LLaVA	7.7	12.3	9.5	84.2
	TimeChat	7.1	50.1	12.4	52.2
Measurement Error	Video-LLaVA	0.0	0.0	0.0	93.7
	TimeChat	6.1	44.2	10.7	57.1
Technique Error	Video-LLaVA	11.1	0.4	0.7	90.9
	TimeChat	2.4	0.4	0.6	89.9
Missing Steps	Video-LLaVA	5.1	3.9	4.4	91.7
	TimeChat	2.2	0.4	12.4	94.3
Temperature Error	Video-LLaVA	0.5	4.6	0.9	89.4
	TimeChat	0.6	6.2	1.2	88.4
Timing Error	Video-LLaVA	6.6	0.6	1.1	96.8
	TimeChat	3.0	17.6	5.1	80.5

Error Category Recognition. In Table 2, we presented evaluation results for the task Error Category Recognition (namely, classify whether a video includes an error of a specific category or not.) formulated as a binary classification problem. We used the standard binary classification metrics to report the evaluation results. Specifically, we employed the following methodology:

- We construct an error category-specific question prompt for each step (refer Fig. 4).
- Using error annotations, we constructed error-category-specific label for each step.
- We processed the generated responses by VLM to obtain the predictions for recipe steps.
- We evaluated the obtained prediction using the labels constructed above. (refer Table 2).

Insights: (1) Video-LLaVa and TimeChat exhibit better performance on different categories of errors. Thus suggesting a VLM ensemble as a natural extension to estimate final predictions. (2) Error Category Recognition is a problem with heavy class imbalance, which can be inferred from the reported scores of accuracy and the F1 metrics in Tab. 2. (3) The low scores indicate the difficulty of the task, and we hope that these numbers will improve with more advanced closed-source VLMs.

Examples: We present category-specific templates and the corresponding examples.

Error Category: Technique Error

VLM: Video-LLaVa

Template: ASSISTANT: To prepare recipe, the person {should, has, ...} to {perform, execute, doing, ...} the step recipe step. Answer with a yes or no, {Did, Does, Has, ...} the person {carefully, precisely, ...} {perform, execute, doing, ...} the recipe step {without spilling, dropping, ...}?

VLM: TimeChat

Template: You are given a cooking video. Please watch the video and answer the following question: To prepare recipe, the person {should, has, ...} to {perform, execute, doing, ...} the step recipe step. {Did, Does, Has, ...} the person {carefully, precisely, ...} {perform, execute, doing, ...} the recipe step? Return the answer in the format of Yes or No.

Question Prompts

Recipe: Cucumber Raita

Step: *Add the chopped or grated cucumber to the whisked curd.*

VLM: Video-LLaVa

Prompt: To prepare Cucumber Raita, the person has to add the chopped or grated cucumber to the whisked curd. Answer with a yes or no, does the person carefully add chopped or grated cucumber to the curd without spilling?

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: To prepare Cucumber Raita, the person has to add chopped or grated cucumber to whisked curd. Does the person carefully add chopped or grated cucumber to the curd? Return the answer in the format of Yes or No.

Recipe: Spiced Hot Chocolate

Step: *Add 1/5 teaspoon cinnamon to the mug.*

VLM: Video-LLaVa

Prompt: To prepare Spiced Hot Chocolate, the person should add 1/5 teaspoon of cinnamon to the mug. Answer with a yes or no, did the person carefully add 1/5 teaspoon of cinnamon to the mug without spilling?

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: To prepare Spiced Hot Chocolate, the person should add 1/5 teaspoon of cinnamon to the mug. Does the person carefully add 1/5 teaspoon of cinnamon to the mug without spilling? Return the answer in the format of Yes or No.

Error Category: Preparation Error

VLM: Video-LLaVa

Template: ASSISTANT: {What, Which, . . . } {tool, ingredient, . . . } is used for recipe step to make recipe ? {Select, Choose, . . . } one of the options: {option 1, option 2 . . . }.

VLM: TimeChat

Template: You are given a cooking video. Please watch the video and answer the following question: {What, Which, . . . } {tool, ingredient, . . . } is used for recipe step to make recipe ? {Select, Choose, . . . } one of the options: {option 1, option 2 . . . }.

Question Prompts

Recipe: Cucumber Raita

Step: *In a mixing bowl, whisk 1 cup of chilled curd until smooth. Use fresh homemade or packaged curd.*

VLM: Video-LLaVa

Prompt: What tool is used for making the chilled fresh homemade or packaged curd smooth in a mixing bowl for making Cucumber Raita? Choose one of the options: (a) whisker, (b) fork, (c) ladle, (d) knife.

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: What tool is used for making the chilled fresh homemade or packaged curd smooth in a mixing bowl for making Cucumber Raita? Choose one of the options: (a) whisker, (b) fork, (c) ladle, (d) knife.

Recipe: Spiced Hot Chocolate

Step: *Microwave the contents of the mug for 1 minute.*

VLM: Video-LLaVa

Prompt: Which tool is used to heat the contents of the mug to make Spiced Hot Chocolate? Choose one of the options: (a) microwave, (b) saucepan, (c) toaster, (d) kettle.

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: Which tool is used to heat the contents of the mug to make Spiced Hot Chocolate? Choose one of the options: (a) microwave, (b) saucepan, (c) toaster, (d) kettle.

Error Category: Order Error

VLM: Video-LLaVa

Template: ASSISTANT: {Did, Is, Does, ...} the person {perform, execute, doing, ...} the step recipe step to {cook, make} recipe ? {Has, Have, ...} the previous recipe step(s) been {completed, performed, ...} before recipe step ?

VLM: TimeChat

Template: You are given a cooking video. Please watch the video and answer the following question: {Did, Is, Does, ...} the person {perform, execute, doing, ...} the step recipe step to {cook, make} recipe ? {Has, Have, ...} the previous recipe step(s) been {completed, performed, ...} before recipe step ? Return the answer in the format of Yes or No.

Question Prompts

Recipe: Cucumber Raita

Step: *Peel the cucumber.*

VLM: Video-LLaVa

Prompt: Did the person peel the cucumber to make Cucumber Raita? Has 1 medium sized cucumber been rinsed before peeling the cucumber?

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: Did the person peel the cucumber to make Cucumber Raita? Has 1 medium sized cucumber been rinsed before peeling the cucumber? Return the answer in the format of Yes or No.

Recipe: Spiced Hot Chocolate

Step: *Heat the contents of the mug for 1 minute and serve.*

VLM: Video-LLaVa

Prompt: Does the person heat the contents of the mug for 1 minute and served to cook Spiced Hot Chocolate? Have the contents of the mug been mixed before heating for 1 minute and serving?

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: Does the person heat the contents of the mug for 1 minute and served to cook Spiced Hot Chocolate? Have the contents of the mug been mixed before heating for 1 minute and serving? Return the answer in the format of Yes or No.

Error Category: Missing Steps

VLM: Video-LLaVa

Template: {Did, Is, Does, . . . } the person {perform, execute, doing, . . . } the step recipe step from the recipe recipe ?

VLM: TimeChat

Template: You are given a cooking video. Please watch the video and answer the following question: {Did, Is} the person {perform, doing} the step recipe step ? Return the answer in the format of Yes or No.

Question Prompts

Recipe: Cucumber Raita

Step: *In a mixing bowl, whisk 1 cup of chilled curd until smooth. Use fresh homemade or packaged curd*

VLM: Video-LLaVa

Prompt: Did the person whisk 1 cup of chilled fresh homemade or packaged curd until smooth while performing the Cucumber Raita recipe?

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: Did the person whisk 1 cup of chilled fresh homemade or packaged curd until smooth? Return the answer in the format of Yes or No.

Recipe: Spiced Hot Chocolate

Step: *Add 2 pieces of chocolate to the mug*

VLM: Video-LLaVa

Prompt: Does the person add 2 pieces of chocolate to the mug while performing Spiced Hot Chocolate recipe?

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: Does the person add 2 pieces of chocolate to the mug? Return the answer in the format of Yes or No.

Error Category: Measurement Error

VLM: Video-LLaVa

Template: ASSISTANT: To {complete, cook, . . . } the recipe recipe, the person should {prepare, do, . . . } the step recipe step. {Does, Did, . . . } the person {measure, weigh} the {ingredient} accurately?

VLM: TimeChat

Template: You are given a cooking video. Please watch the video and answer the following question: To {complete, cook, . . . } the recipe recipe, the person should {prepare, do, . . . } the step recipe step. {Does, Did, . . . } the person {measure, weigh} the {ingredient} accurately? Return the answer in the format of Yes or No.

Question Prompts

Recipe: Cucumber Raita

Step: *Add 1/4 teaspoon of salt to the bowl.*

VLM: Video-LLaVa

Prompt: To make the recipe Cucumber Raita, the person should add 1/4 teaspoon of salt to the bowl. Does the person measure 1/4 teaspoon of salt accurately?

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: To make the recipe Cucumber Raita, the person should add 1/4 teaspoon of salt to the bowl. Does the person measure 1/4 teaspoon of salt accurately? Return the answer in the format of Yes or No.

Recipe: Spiced Hot Chocolate

Step: *Add 1/5 teaspoon of cinnamon to the mug.*

VLM: Video-LLaVa

Prompt: To cook the recipe Spiced Hot Chocolate, the person should add 1/5 teaspoon of cinnamon to the mug. Does the person measure 1/5 teaspoon of cinnamon correctly?

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: To cook the recipe Spiced Hot Chocolate, the person should add 1/5 teaspoon of cinnamon to the mug. Does the person measure 1/5 teaspoon of cinnamon correctly? Return the answer in the format of Yes or No.

Error Category: Temperature Error

VLM: Video-LLaVa

Template: ASSISTANT: {While, When ...} the person is {performing, executing, ...} the step recipe step from the recipe recipe. Is any heating involved? if yes, then did the person adhere to the {low, medium, high} {heating, power level} settings of {microwave, stove}.

VLM: TimeChat

Template: You are given a cooking video. Please watch the video and answer the following question: {While, When ...} the person is {performing, executing, ...} the step recipe step from the recipe recipe. Is any heating involved? if yes, then did the person adhere to the {low, medium, high} {heating, power level} settings of {microwave, stove}. Return the answer in the format of Yes or No.

Question Prompts

Recipe: Cucumber Raita

Step: *Peel a cucumber*

VLM: Video-LLaVa

Prompt: While the person is peeling a cucumber for making Cucumber Raita. Is any heating involved? If yes, did the person adhere to the heating settings?

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: While the person is peeling a cucumber for making Cucumber Raita. Is any heating involved? If yes, did the person adhere to the heating settings? Return the answer in the format of Yes or No.

Recipe: Spiced Hot Chocolate

Step: *Heat the contents of the mug for 1 minute and serve.*

VLM: Video-LLaVa

Prompt: When the person has to heat the contents of the mug for 1 minute and serve for cooking Spiced Hot Chocolate, is any heat required? If yes, did the person adhere to the high heat setting of microwave?

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: When the person has to heat the contents of the mug for 1 minute and serve for cooking Spiced Hot Chocolate, is any heat required? If yes, did the person adhere to the high heat setting of microwave? Return the answer in the format of Yes or No.

Error Category: Timing Error

VLM: Video-LLaVa

Template: To {cook, make, . . . } recipe , the person {should, has . . . } to recipe step . {Should, Does} the person recipe step {perform, make, . . . } for a {specific, certain} time?

VLM: TimeChat

Template: You are given a cooking video. Please watch the video and answer the following question: To {cook, make, . . . } recipe , the person {should, has . . . } to recipe step . {Should, Does} the person recipe step {perform, make, . . . } for a {specific, certain} time? Return the answer in the format of Yes or No.

Question Prompts

Recipe: Butter Corn Cup

Step: *Microwave the corn for 2 minutes.*

VLM: Video-LLaVa

Prompt: To make Butter Corn Cup, the person should microwave the corn for 2 minutes. Did the person microwave the corn for 2 minutes?

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: To make Butter Corn Cup, the person should microwave the corn for 2 minutes. Did the person microwave the corn for 2 minutes? Return the answer in the format of Yes or No.

Recipe: Spiced Hot Chocolate

Step: *Heat the contents of the mug for 1 minute and serve.*

VLM: Video-LLaVa

Prompt: To cook Spiced Hot Chocolate, the person should heat the contents of the mug for 1 minute and serve. Does the person heat the contents of the mug for 1 minute before serving?

VLM: TimeChat

Prompt: You are given a cooking video. Please watch the video and answer the following question: To cook Spiced Hot Chocolate, the person should heat the contents of the mug for 1 minute and serve. Does the person heat the contents of the mug for 1 minute before serving? Return the answer in the format of Yes or No.

B.2 Anomaly Detection

We used anomaly detection methods to classify each frame in each video as either normal or abnormal, where the latter is defined as an instance that deviates from the expected behaviour (the frame where participants made errors). Specifically, we used two self-supervised anomaly detection methods from the literature, self-supervised masked convolutional transformer block (SSMCTB) [42] and self-supervised predictive convolutional attentive block (SSPCAB) [61], and trained them on top of ResNet-50 [25], where the latter serves as a neural, image-based feature extractor. Both models were trained using reconstruction loss [42]. We used normal recordings for training and both normal and error recordings for testing. We evaluated the benchmark models using the frame-level area under the curve (AUC) and Equal Error Rate (EER) scores. Table 3 shows the results. We observe that SSMCTB is slightly better than SSPCAB. The AUC scores displayed in this context demonstrate only marginal improvement over random chance. This emphasizes the difficulty of the task and underscores the necessity for specialized approaches to recognize errors in a self-supervised manner.

Table 3: **Anomaly Detection**

Method	AUC(%)	EER (%)
SSMCTB [42]	50.65	49.65
SSPCAB [61]	50.25	49.74

B.3 Supervised Error Recognition

Task. We set up the error recognition task (namely, given a video segment, classify it either as error or normal) as a supervised binary classification problem. The presence of a variety of errors (that are both cascading and non-cascading in nature across the duration of the recording) makes solving this task particularly challenging. We use error annotations and mark a segment as *normal* if the corresponding step was performed correctly; else, we mark it as an *error*.

Features. We obtained features from pre-trained video recognition models, namely (1) Slowfast, (2) X3D, (3) Omnivore, (4) 3D Resnet, and (5) Imagebind. Since the feature extractors require fixed-sized inputs (they are neural networks), we divided each video segment into contiguous 1-second sub-segments. The video segment may not always be perfectly divisible by 1 second as the last sub-segment might be shorter than 1 second. To make it uniform, we used zero padding; namely, we added zeros at the end of the sub-segment and extended its duration to 1 second to extract its features.

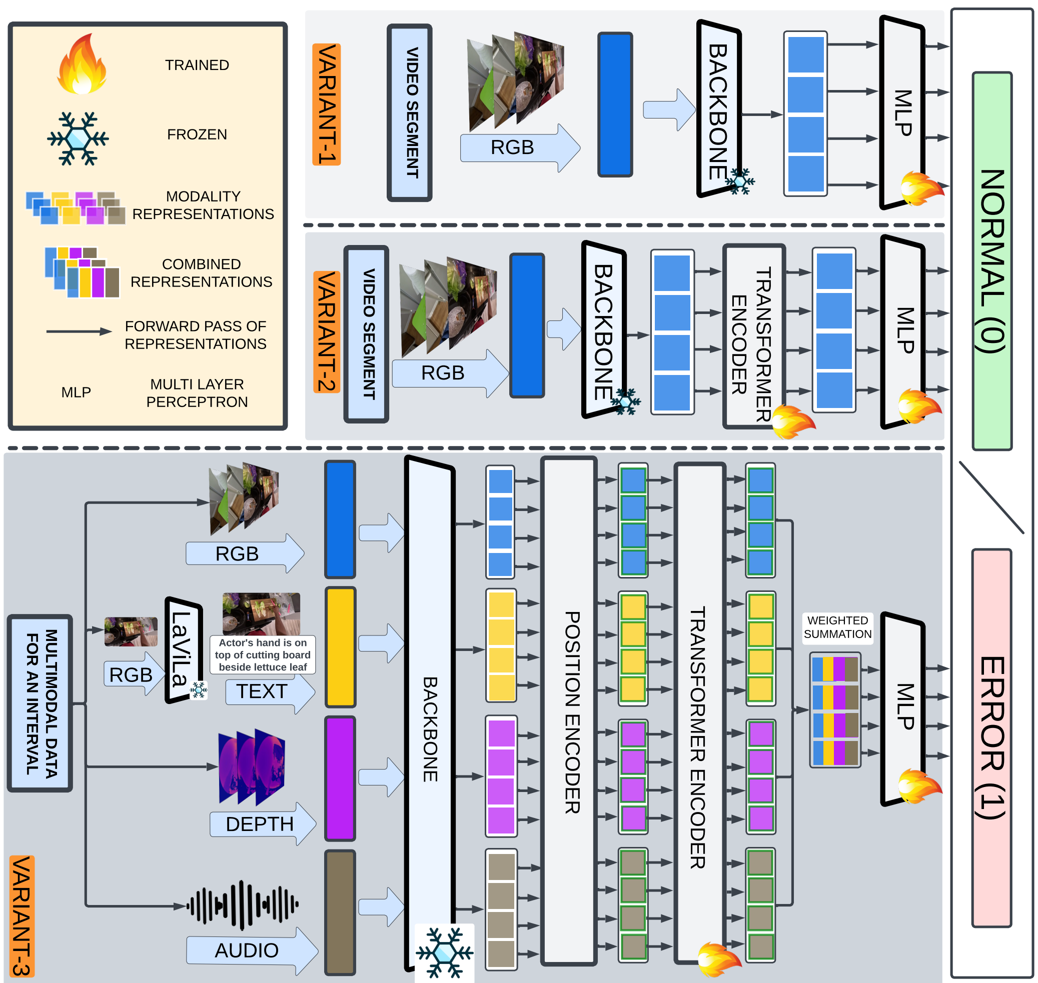

Figure 5: **SupervisedER** architectures of 3 baselines.

Models. We proposed three architectural variants as baselines for Supervised Error Recognition(Fig. 5). During training, we assigned a segment's (recipe step) class label to all its 1-second sub-segments (for which features are extracted). Thus yielding the proposed splits' train, validation, and test subsets of data, which are used to learn our proposed variants of the baselines. During inference, we again divided each video segment into 1-second sub-segments and, after applying any necessary zero-padding, designated the class of the segment as the majority class of its sub-segments.

Variant-1: Below are the rough steps we followed to train our $\mathcal{V}_1$ supervised error recognition models.

- **Dataset:** We used two proposed splits, namely, Step ($\mathcal{S}$) and Recordings ($\mathcal{R}$) for training.
- **Model:** We trained our models using a Multi-Layer Perceptron (MLP) Head that includes one hidden layer. The size of this layer depends on the feature dimensions from the pre-trained video recognition models. The hidden layer is followed by a single sigmoid node.
- **Training:** We trained these models on the training subset and fine-tuned the hyperparameters using the validation subsets of the proposed splits. We maintained a uniform minibatch size of 512 instances. We employed ReLU activation functions in the hidden layers and trained using the PyTorch [53] on a single NVIDIA A40 GPU. We employed the Adam optimizer [36] for training and a learning rate of 0.001. Due to the inherent class imbalance of the constructed dataset, we used standard Binary Cross Entropy Loss (BCE Loss) with a weight of 1.5 for the positive classes. We trained all our models for 50 epochs.
- **Evaluation:** We selected the model that performed best on the validation set, evaluated it on the test set, and presented the evaluation results in the main paper.

Variant-2: Below are the rough steps we followed to train our $\mathcal{V}_2$ supervised error recognition models.

- **Dataset:** We used two proposed splits, namely, Step ($\mathcal{S}$) and Recordings ($\mathcal{R}$) for training.
- **Model:** We leveraged the strengths of transformers that facilitate using variable length inputs and allow representing time via positional encodings to train baseline error recognition models. Specifically, instead of generating predictions for 1-second sub-segments independently, we pass the sub-segment features through a transformer encoder to learn representations that are contextually aware of the entire video segment of the recipe step. Finally, we pass these representations of 1-second sub-segments through a Multi-Layer Perceptron (MLP) head that includes one hidden layer (whose size is determined by the feature dimensions of the pre-trained video recognition models) followed by a single sigmoid node.
- **Training:** We trained these models on the training subset and fine-tuned the hyperparameters using the validation subsets of the proposed splits. We maintained a uniform minibatch size of 1 video segment corresponding to a step. We trained using the PyTorch [53] on a single NVIDIA A40 GPU. We employed the Adam optimizer [36] for training and a learning rate of 1e-5. To address the inherent class imbalance of the dataset, we used standard BCE Loss with a weight of 1.5 for the positive classes and trained all our models for 50 epochs.
- **Evaluation:** We selected the model that performed best on the validation set, evaluated it on the test set, and presented the evaluation results in the main paper.

Variant-3: Below are the rough steps we followed to train our $\mathcal{V}_3$ supervised error recognition models.

- **Dataset:** We used two proposed splits, namely, Step ($\mathcal{S}$) and Recordings ($\mathcal{R}$) for training.
- **Model:** We leveraged the strengths of transformers that facilitate using variable length inputs from multiple modalities of data and allow representing time via positional encodings to train baseline error recognition models. Specifically, instead of generating predictions for a single RGB modality data, we pass the sub-segment features corresponding to RGB, audio, text and depth modalities through a transformer encoder to learn unified representations that are contextually aware of the entire video segment of the recipe step. Finally, we pass these unified representations of multiple modalities that correspond to sub-segments of the data through an MLP head that includes one hidden layer (whose size is determined by the features of the pre-trained video recognition models) followed by a single sigmoid node.
- **Training:** We trained these models on the training subset and fine-tuned the hyperparameters using the validation subsets of the proposed splits. We maintained a uniform minibatch size of 1 video segment corresponding to a step. We trained using the PyTorch [53] on a single NVIDIA A40 GPU. We employed the Adam optimizer [36] for training and a learning rate

of 5e-5. To address the inherent class imbalance of the dataset, we used standard BCE Loss with a weight of 1.5 for the positive classes and trained all our models for 50 epochs.

– **Evaluation:** We selected the model that performed best on the validation set, evaluated it on the test set, and presented the evaluation results in the main paper.

Table 4: **SupervisedER** evaluation results of baselines. Variant type ($\mathcal{V}_{\#}$), Modality of features (M), Video (V), Audio (A), Depth (D), and Text (T).

Split	Backbone	$\mathcal{V}_{\#}$	Modality	Acc	P	R	F1	AUC
$\mathcal{S}$	Omnivore	$\mathcal{V}_1$	V	71.09	66.07	14.86	24.26	75.7
		$\mathcal{V}_2$	V	69.96	51.56	59.84	55.39	75.7
	Slowfast	$\mathcal{V}_1$	V	33.54	31.88	90.6	47.16	63.06
		$\mathcal{V}_2$	V	68.09	47.69	24.9	32.72	67.18
	X3D	$\mathcal{V}_1$	V	68.34	48	19.28	27.51	60.19
		$\mathcal{V}_2$	V	67.83	42.86	9.64	15.74	61.5
	3DResnet	$\mathcal{V}_1$	V	63.45	42.9	52.21	47.1	66.16
		$\mathcal{V}_2$	V	61.58	41.47	56.63	47.88	64.5
	ImageBind	$\mathcal{V}_3$	V	62.78	38.46	32.13	35.01	57.03
			A	43.86	32.26	72.69	44.69	52.23
			V, A	63.28	40.76	38.96	39.84	53.87
			V, A, T	68.79	42.76	41.96	42.36	61.1
			V, A, D, T	69.4	50.75	48.96	49.84	70.41
$\mathcal{R}$	Omnivore	$\mathcal{V}_1$	V	59.76	45.31	58.09	50.91	63.03
		$\mathcal{V}_2$	V	62.3	46.55	33.61	39.04	62.27
	Slowfast	$\mathcal{V}_1$	V	60.06	40.82	24.9	30.93	56.89
		$\mathcal{V}_2$	V	57.82	41.67	43.57	42.6	59.83
	X3D	$\mathcal{V}_1$	V	54.69	39.6	49.79	44.12	54.66
		$\mathcal{V}_2$	V	54.25	40.78	60.58	48.75	56.58
	3DResnet	$\mathcal{V}_1$	V	41.88	37.65	94.19	53.79	62.42
		$\mathcal{V}_2$	V	57.82	43.56	58.92	50.09	59.22
	ImageBind	$\mathcal{V}_3$	V	37.56	36.14	96.27	52.55	54.1
			A	39.05	35.67	89.72	50.54	54.8
			V, A	54.25	40.98	62.24	49.42	55.25
			V, A, T	64.08	44.15	58.01	50.13	58.35
			V, A, D, T	63.87	49.57	64.4	56.02	65.25

B.4 Supervised Early Error Recognition

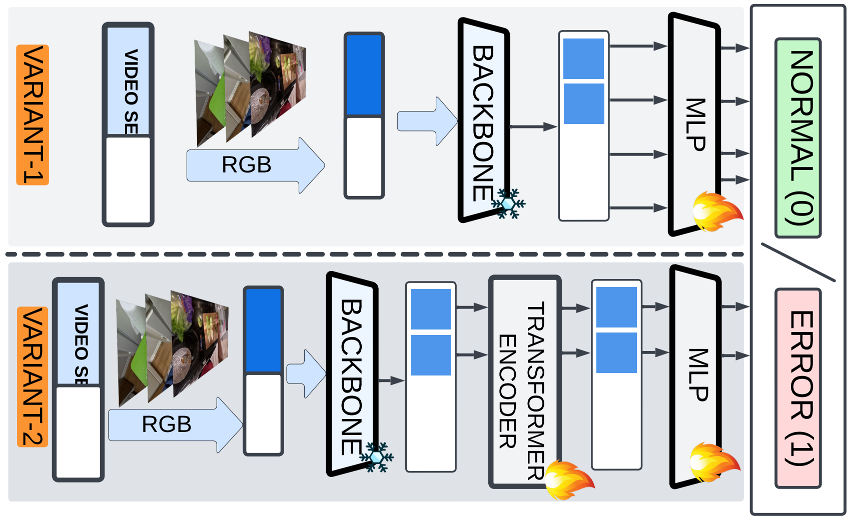

Figure 6: **Supervised Early Error Recognition** architectures of baselines.

Task. We set up the early error recognition task (namely, given only the first half of the video segment corresponding to a step, classify it either as error or normal) as a supervised binary classification problem. Since the model only processes the first half of the video segment, which mainly showcases the pre-conditions for an action, error recognition in this context involves ***anticipating potential errors*** that may arise from deviations in the pre-conditions of the action. Thus, early error recognition is an extremely hard setting and the presence of a variety of errors (that are both cascading and non-cascading in nature across the duration of the recording) makes it more challenging.

Features. We obtained features from pre-trained video recognition models, namely (1) Slowfast, (2) X3D, (3) Omnivore, (4) 3D Resnet. Since the feature extractors require fixed-sized inputs (they are neural networks), we divided each video segment into contiguous 1-second sub-segments. The video segment may not always be perfectly divisible by 1 second as the last sub-segment might be shorter than 1 second. To make it uniform, we used zero padding; namely, we added zeros at the end of the sub-segment and extended its duration to 1 second to extract its features.

Models. We proposed two architectural variants as baselines for Supervised Early Error Recognition(Fig. 6). During training, we assigned a segment's (recipe step) class label to all its observed 1-second sub-segments (first half of the video segments). Thus yielding the proposed splits' train, validation, and test subsets of data, which are used to train our baselines. During inference, we again divided each partially observed video segment into 1-second sub-segments and assigned the class label of the partially observed video segment as the majority class of observed sub-segments.

<u>Variant-1</u>: Rough steps we followed to train our $\mathcal{V}_1$ supervised early error recognition models.

- **Dataset:** We used two proposed splits, namely, Step ($\mathcal{S}$) and Recordings ($\mathcal{R}$) for training.
- **Model:** We trained our models using a Multi-Layer Perceptron (MLP) Head that includes one hidden layer. The size of this layer depends on the feature dimensions from the pre-trained video recognition models. The hidden layer is followed by a single sigmoid node.
- **Training:** We trained these models on the training subset and fine-tuned the hyperparameters using the validation subsets of the proposed splits. We maintained a uniform minibatch size of 512 instances. We employed ReLU activation functions in the hidden layers and trained using the PyTorch [53] on a single NVIDIA A40 GPU. We employed the Adam optimizer [36] for training and a learning rate of 1e-4. Due to the inherent class imbalance of the

constructed dataset, we used standard Binary Cross Entropy Loss (BCE Loss) with a weight of 1.5 for the positive classes. We trained all our models for 50 epochs.

- **Evaluation:** We selected the model that performed best on the validation set, evaluated it on the test set, and presented the evaluation results in Table 5.

Variant-2: Rough steps we followed to train our $\mathcal{V}_2$ supervised error recognition models.

- **Dataset:** We used two proposed splits, namely, Step ($\mathcal{S}$) and Recordings ($\mathcal{R}$) for training.
- **Model:** Instead of generating predictions for 1-second sub-segments independently, we pass the sub-segment features through a transformer encoder to learn representations that are contextually aware of the entire video segment of the recipe step. Finally, we pass these representations of 1-second sub-segments through a Multi-Layer Perceptron (MLP) head that includes one hidden layer (whose size is determined by the feature dimensions of the pre-trained video recognition models) followed by a single sigmoid node.
- **Training:** We trained these models on the training subset and fine-tuned the hyperparameters using the validation subsets of the proposed splits. We maintained a uniform minibatch size of 1 video segment corresponding to a step. We trained using the PyTorch [53] on a single NVIDIA A40 GPU. We employed the Adam optimizer [36] for training and a learning rate of 1e-5. To address the inherent class imbalance of the dataset, we used standard BCE Loss with a weight of 1.5 for the positive classes and trained all our models for 50 epochs.
- **Evaluation:** We selected the model that performed best on the validation set, evaluated it on the test set, and presented the evaluation results in Table 5.

Table 5: **SupervisedEER** evaluation results of baselines. Variant type ($\mathcal{V}_{\#}$), Modality of features (M), Video (V), Audio (A), Depth (D), and Text (T).

Split	Backbone	$\mathcal{V}_{\#}$	Modality	Acc	P	R	F1	AUC
$\mathcal{S}$	Omnivore	$\mathcal{V}_1$	V	69.59	75	3.61	6.9	72.9
		$\mathcal{V}_2$	V	69.21	50	1.2	2.34	73.48
	Slowfast	$\mathcal{V}_1$	V	67.96	47.15	23.29	31.18	67.23
		$\mathcal{V}_2$	V	69.09	50	1.6	3.1	62.72
	X3D	$\mathcal{V}_1$	V	65.96	41.73	23.29	29.9	59.27
		$\mathcal{V}_2$	V	68.71	45.45	2.01	3.85	56.5
	3DResnet	$\mathcal{V}_1$	V	68.46	47.89	13.65	21.25	63.79
		$\mathcal{V}_2$	V	68.84	50	0.4	0.8	60.76
$\mathcal{R}$	Omnivore	$\mathcal{V}_1$	V	64.08	50	2.07	3.98	64.73
		$\mathcal{V}_2$	V	64.08	50	0.41	0.82	65.43
	Slowfast	$\mathcal{V}_1$	V	63.79	33.33	0.83	1.62	53.11
		$\mathcal{V}_2$	V	63.93	42.86	1.24	2.42	53.38
	X3D	$\mathcal{V}_1$	V	63.34	46.27	12.86	20.13	55.24
		$\mathcal{V}_2$	V	63.64	28.57	0.83	1.61	56.84
	3DResnet	$\mathcal{V}_1$	V	63.19	43.75	8.71	14.53	53.47
		$\mathcal{V}_2$	V	63.04	37.04	4.15	7.46	54.73

Remark: We observed that there is a significant drop in performance metrics of all our models compared to the Supervised Error Recognition task. We also note that our $\mathcal{V}_2$ models exhibited lower performance than our $\mathcal{V}_1$ models. We attribute this drop to the extremely noisy signal (due to the observation of only the pre-conditions of actions) used to recognize errors in the recordings.

B.5 Multi-Step Localization

Multi-Step localization (MSL) entails both recognizing and localization of steps within a procedural activity. For this task, we leverage features extracted from pre-trained video recognition models and train an ActionFormer head to manage the processes of step recognition and localization. In the main text, we detailed the experimental evaluations of MSL and Robust MSL. Additionally, we trained models using features extracted with Omnivore as the backbone for video segments of 1-second, 3-second, and 4-second lengths, and we present the results in Table 6. We observed a performance enhancement in the model as the length of the video segments used for feature extraction increased.

Table 6: **MSL** using Omnivore features corresponding to video segments of varying lengths

$\mathcal{B}$	$\mathcal{D}$	$\mathcal{I}_t = 0.1$			$\mathcal{I}_t = 0.3$			$\mathcal{I}_t = 0.5$		
		mAP	**R@1**	**R@5**	**mAP**	**R@1**	**R@5**	**mAP**	**R@1**	**R@5**
$\mathcal{O}_{1s}$	$\mathcal{E}$	67.51	64.45	62.31	85.32	82.82	78.11	38.32	36.54	33.41
	$\mathcal{P}$	**75.96**	**73.35**	**70.34**	**92.14**	**90.51**	**88.24**	**45.82**	**44.12**	**41.16**
	$\mathcal{R}$	73.71	71.45	68.14	92.08	89.82	86.38	42.76	40.52	37.19
$\mathcal{O}_{3s}$	$\mathcal{E}$	72.99	70.05	66.57	86.03	83.68	81.02	43.47	41.83	38.87
	$\mathcal{P}$	**78.63**	**76.96**	**74.61**	**93.27**	**91.23**	**89.18**	**50.25**	**48.54**	**44.85**
	$\mathcal{R}$	76.82	74.90	71.94	91.33	89.61	88.11	49.23	47.84	44.76
$\mathcal{O}_{4s}$	$\mathcal{E}$	71.85	69.79	64.93	88.12	86.33	83.15	43.13	41.54	38.95
	$\mathcal{P}$	**79.33**	**77.39**	**74.24**	**93.46**	**91.67**	**89.95**	**50.69**	**49.49**	**46.19**
	$\mathcal{R}$	78.61	76.59	73.81	93.04	90.99	88.80	50.24	48.62	45.64

Table 7: **MSL** evaluation results.

$\mathcal{B}$	$\mathcal{D}$	$\mathcal{I}_t = 0.1$			$\mathcal{I}_t = 0.3$			$\mathcal{I}_t = 0.5$		
		mAP	**R@1**	**R@5**	**mAP**	**R@1**	**R@5**	**mAP**	**R@1**	**R@5**
3D Resnet	$\mathcal{E}$	25.98	54.82	77.59	23.75	48.38	72.19	19.59	38.44	61.87
	$\mathcal{P}$	29.29	63.07	88.96	27.71	56.60	84.75	23.21	46.79	76.86
	$\mathcal{R}$	29.39	61.14	85.41	27.89	55.82	82.17	23.97	46.54	73.29
Slowfast	$\mathcal{E}$	27.68	55.73	77.45	25.51	48.98	70.90	21.09	37.82	60.58
	$\mathcal{P}$	32.77	63.22	90.43	31.21	58.82	86.82	27.25	50.70	79.49
	$\mathcal{R}$	32.90	63.97	89.29	31.47	59.26	85.32	27.89	51.62	77.27
VideoMAE	$\mathcal{E}$	28.12	51.76	73.00	26.38	46.16	67.87	21.35	37.12	57.81
	$\mathcal{P}$	38.86	64.86	84.05	37.41	60.32	80.63	32.24	51.46	71.88
	$\mathcal{R}$	37.44	63.08	80.90	35.11	57.30	77.38	30.76	49.19	69.43
Omnivore	$\mathcal{E}$	40.40	67.51	87.69	38.32	62.31	82.82	33.41	53.01	72.85
	$\mathcal{P}$	48.16	75.96	93.41	45.82	70.34	90.51	41.16	62.00	84.73
	$\mathcal{R}$	44.81	73.71	93.34	42.76	68.14	89.82	37.19	56.93	81.86

Table 8: **RobustMSL** evaluation results.

$\mathcal{B}$	$\mathcal{D}$	$\mathcal{T}$	$\mathcal{I}_t = 0.1$			$\mathcal{I}_t = 0.3$			$\mathcal{I}_t = 0.5$		
			mAP	**R@1**	**R@5**	**mAP**	**R@1**	**R@5**	**mAP**	**R@1**	**R@5**
VideoMAE	$\mathcal{E}$	$\mathcal{T}_n$	24.44	38.22	52.48	22.97	34.77	49.51	18.67	28.57	42.68
		$\mathcal{T}_e$	7.53	13.54	20.52	6.93	11.4	18.36	5.63	8.55	15.13
	$\mathcal{P}$	$\mathcal{T}_n$	26.78	37.43	46.28	25.68	34.79	44.6	22.02	29.43	39.81
		$\mathcal{T}_e$	16.98	27.43	37.76	16.46	25.53	36.03	14.64	22.03	32.07
	$\mathcal{R}$	$\mathcal{T}_n$	26.27	37.15	46.93	24.71	34.06	45.03	21.51	29.36	40.44
		$\mathcal{T}_e$	15.43	25.94	33.97	14.44	23.23	32.35	12.96	19.83	28.99
Omnivore	$\mathcal{E}$	$\mathcal{T}_n$	34.65	47.91	60.63	33.06	44.77	58.36	28.59	38.38	51.9
		$\mathcal{T}_e$	12.51	19.6	27.06	11.66	17.54	24.45	9.94	14.63	20.96
	$\mathcal{P}$	$\mathcal{T}_n$	32.5	44.45	52.47	31.13	41.53	50.91	28.39	37.03	47.97
		$\mathcal{T}_e$	21.28	31.51	40.93	20.12	28.81	39.6	18.08	24.96	36.77
	$\mathcal{R}$	$\mathcal{T}_n$	30.22	42.43	52.11	28.94	39.47	50.49	25.15	32.65	46.51
		$\mathcal{T}_e$	19.54	31.28	41.24	18.4	28.66	39.33	16.27	24.28	35.35

Extended Analysis. In Tables 7 and 8, we note that when data is split by environments, with the test set comprising new environments, the models, in general, struggle to recognize the steps performed in the videos. As we increase thresholded Intersection Over Union ($\mathcal{I}_t$), we observe a drop in the performance of the models, thus signifying the low confidence in the prediction of the current steps.

B.6 Procedure Learning

Given long, untrimmed videos of procedural activities where the sequences of steps can be performed in multiple orders, self-supervised procedure learning entails the identification of relevant frames across videos of activity and the estimation of sequential steps required to complete the activity. Thus, the task entails the identification of key steps and their sequence to complete an activity. To benchmark procedure learning, we used normal recordings from our dataset and assessed the performance of recently proposed methods [4, 15]. In the main text, we presented results when evaluated on only 5 recipes. Here, in Table 9, we present the evaluation results on all 24 recipes. The results in Table 9 showcase the performance of models trained using methods $\mathcal{M}_1$ [15] and $\mathcal{M}_2$ [4]. Where $\mathcal{M}_1$ employs Cycleback Regression Loss ($\mathcal{C}$) and $\mathcal{M}_1$ employs a combination of both Cycleback Regression Loss ($\mathcal{C}$) and Contrastive - Inverse Difference Moment Loss ($\mathscr{C}$). It is important to note that we only train embedder networks using these loss functions and maintain the Pro-Cut Module (PCM) for assigning frames to key steps. In Table 9, $\mathcal{P}$ represents precision, R represents recall, and I represents IOU.

Table 9: **Self-Supervised Procedure Learning** evaluation results on the selected 24 recipes.

Recipe	Random			$\mathcal{M}_1$			$\mathcal{M}_2$		
	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{I}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{I}$	$\mathcal{P}$	$\mathcal{R}$	$\mathcal{I}$
BlenderBananaPancakes	7.40	3.83	2.26	12.65	9.50	5.16	15.54	9.96	5.72
BreakfastBurritos	9.66	4.04	2.59	18.72	11.46	6.77	16.58	10.77	5.87
BroccoliStirFry	4.21	3.81	1.73	9.92	9.11	3.93	8.20	8.10	3.85
ButterCornCup	8.37	3.91	2.16	13.82	11.85	5.79	15.07	12.30	5.82
CapreseBruschetta	9.34	3.96	2.52	25.55	12.89	7.52	20.53	9.09	5.59
CheesePimiento	9.10	3.87	2.41	19.74	10.48	6.44	17.49	10.32	6.26
Coffee	6.54	3.87	2.17	13.68	9.91	5.49	15.76	10.25	5.63
CucumberRaita	8.90	3.64	2.44	13.58	7.92	5.14	16.15	9.97	6.09
DressedUpMeatballs	7.28	3.80	2.26	15.20	10.80	6.05	17.59	10.27	5.81
HerbOmeletWithFriedTomatoes	6.82	4.05	1.98	14.66	14.98	5.50	14.64	11.34	6.29
MicrowaveEggSandwich	8.81	3.98	2.61	16.25	10.44	6.16	19.16	11.29	6.99
MicrowaveFrenchToast	9.03	3.74	2.49	16.82	7.90	5.07	17.31	8.82	5.66
MicrowaveMugPizza	7.53	3.90	2.38	12.82	9.78	5.27	12.69	9.18	5.18
MugCake	5.45	4.00	2.12	16.12	12.95	6.87	10.32	8.85	4.40
PanFriedTofu	5.35	3.97	1.54	8.86	10.39	3.75	9.34	12.44	3.87
Pinwheels	6.54	4.28	2.13	13.58	11.96	5.92	16.08	13.06	7.05
Ramen	6.85	4.12	1.87	11.09	9.97	4.48	12.90	10.92	5.07
SautedMushrooms	6.08	3.81	2.02	15.06	12.22	6.16	19.54	13.83	7.42
ScrambledEggs	4.74	3.95	1.89	11.11	11.08	5.27	11.70	10.96	5.27
SpicedHotChocolate	14.08	3.82	3.09	29.82	10.58	8.49	29.79	11.04	8.74
SpicyTunaAvocadoWraps	6.25	3.90	2.21	15.62	10.52	5.67	12.47	9.61	5.25
TomatoChutney	5.45	3.89	1.85	12.25	10.68	5.42	12.25	10.68	5.42
TomatoMozzarellaSalad	10.88	3.91	2.38	19.77	10.21	6.01	19.20	10.48	5.96
Zoodles	7.91	4.08	2.22	18.32	12.80	6.37	18.32	12.80	6.37
Average	**7.61**	**3.92**	**2.22**	**15.62**	**10.85**	**5.78**	**15.78**	**10.68**	**5.82**

C Data

C.1 Data Collection Planning

Our objective is to capture data that aids in detecting, segmenting, and analyzing errors that occur during the execution of long procedural tasks. To accomplish this, we need to address the following:

1. **What to record:** Specifically, select the domain and tasks (such as recipes).
2. **How to record:** Choice of appropriate sensors and development of data capturing system.
3. **Whom to record:** This entails participant selection and training.

C.1.1 What to record?

Current procedural activity datasets encompass recorded and curated ones from crowd-sourced online platforms. Amongst the recorded datasets, Breakfast [37], 50Salads [72], CMU-MMAC [13], and GTEA [20] capture people performing cooking activities, and Assembly-101 [18], EPIC-TENTS [31] and MECCANO [58] capture people performing activities related to assembly of toys, tents and lego blocks, respectively. Curated datasets like COIN [74], CrossTask [86], and HowTo100M [46] encompass a wide variety of activities from different domains. We introduced a new perspective on understanding procedural activities from the lens of errors made while performing procedural tasks. We embark on an investigation into this new idea by choosing cooking as the domain of interest. This careful choice stems from the fact that cooking activities often encompass complex procedures and provide an opportunity to capture a plethora of potential, predominantly benign errors.

Table 10: Selected recipes categorized based on the type of required heating instrument.

Heating Instrument	**Recipe**	**Heating Instrument**	**Recipe**
Kettle	Coffee	Nothing	Pinwheels
Microwave	Breakfast Burritos		Spicy Tuna Avocado Wraps
	Butter Corn Cup		Tomato Mozzarella Salad
	Cheese Pimiento	Pan	Blender Banana Pancakes
	Dressed Up Meatballs		Broccoli Stir Fry
	Microwave Egg Sandwich		Caprese Bruschetta
	Microwave French Toast		Herb Omelet with Fried Tomatoes
	Microwave Mug Pizza		Pan Fried Tofu
	Mug Cake		Sauteed Mushrooms
	Ramen		Scrambled Eggs
	Spiced Hot Chocolate		Tomato Chutney
Nothing	Cucumber Raita		Zoodles

Recipes & Task Graphs We have carefully selected 24 diverse recipes from WikiHow (Table 10) that represent various cuisines and require different culinary tools during preparation. Each recipe in our selected set can be subdivided into several atomic steps, where each step involves performing a specific sub-task in the recipe. In general, most recipes available on the web list these sub-tasks in a specific order. However, common sense tells us that each recipe can often be described by a partial order over the sub-tasks rather than a total order. More formally, we use a task graph to represent the partial order over the steps. Each node in the task graph corresponds to a step, and a directed edge between node i and node j denotes that step i must be done before step j (namely i is a pre-condition of j). For our selected recipes, the corresponding task graphs are directed acyclic graphs, and therefore a topological sort over them is a valid execution of the recipe. Our task graphs also include two dummy nodes, "START" and "END", which denote the start and end of recipes, respectively and ensure that our task graphs always have one start node and one terminal node.

To simplify the complexity of a recipe, we have adopted a technique that uses a flow graph structure [79] to represent the dependencies between steps (think of it like a flowchart but designed for recipes). This approach helps us establish a precise connection between actions and their consequences. Using an action-centric graph, we emphasize the steps involved in the procedure and illustrate the sequence of operations in an easy-to-understand manner. Each action influences the subsequent ones, effectively demonstrating the interdependencies between tasks. Figure 8 presents an example of a task graph.

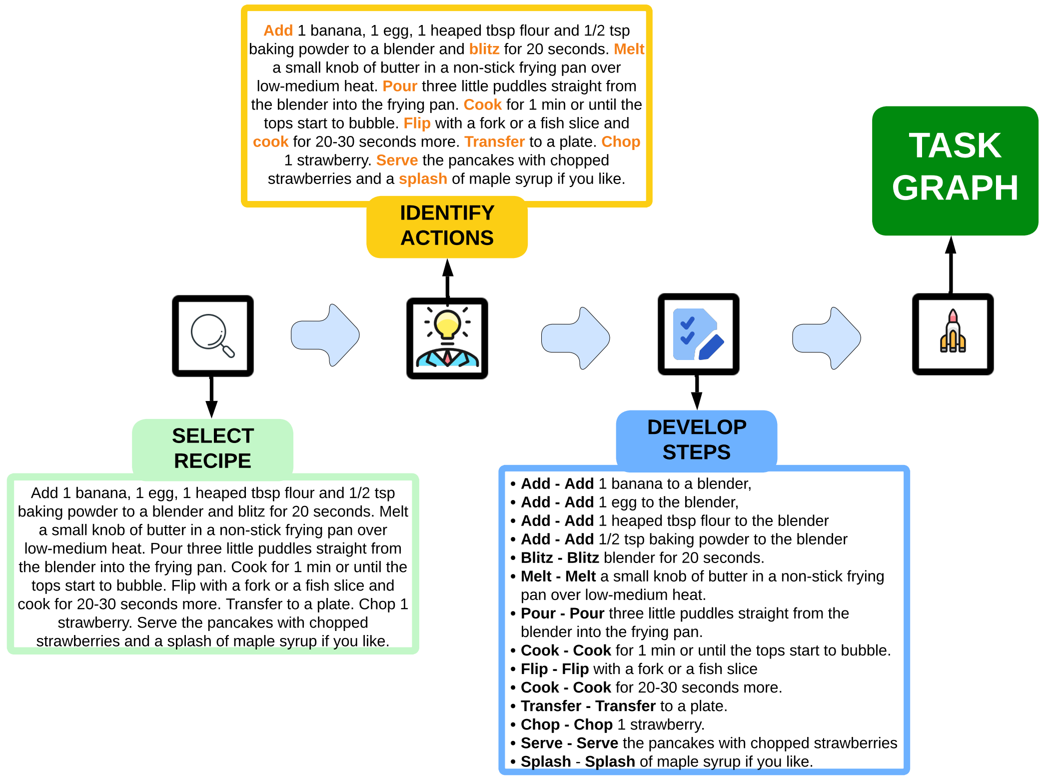

Figure 7: **TaskGraphGeneration.** This figure demonstrates the four-step process used to create an action-centric graph for a recipe, using an example. Given the recipe text as shown in *select recipe*, we identify and mark all the actions necessary for the execution of the recipe as shown in *identify actions*. Once these actions are identified, we develop them into steps (as shown in *develop steps*) ensuring each step encompasses only one of the previously identified actions. These steps are used to construct an action-centric graph for the recipe resulting in a structure as depicted in Figure 8

We illustrate the process we used to convert a recipe to a task graph using the recipe *Blender Banana Pancakes* (see figures 7 and 8 for a visual guide). Given the recipe description, we first identify all the actions necessary to complete the recipe and develop steps based on the identified actions, where each step contains only one among the identified actions, as shown in figure 7. After we develop steps, we use a relationship annotation tool[3] to represent the implicit order constraints amongst the developed steps. The creation of action-centric graphs serves multiple purposes. These graphs can be utilized to prepare recipe scripts with various orders while still strictly adhering to the constraints present in the graph. Moreover, given a recording, the graph can be used to verify if the individual followed the correct sequence of actions based on the inherent graph structure.ipe *Blender Banana Pancakes*, the developed steps from 7, when represented as an action-centric graph, result in figure 8.

In the future, we envision using our dataset to construct more fine-grained task graphs where the meaning of the steps is taken into account and how the step changes the environment (post-condition for a step). In literature, different methods have been proposed to illustrate procedural activities using task graphs and their variations, such as FlowGraphs [79], Recipe Programs [52], ConjugateTaskGraphs [24], and ActionDynamicTaskGraphs [45] and our dataset can be used to learn these task graphs in an unsupervised manner (or one can use the semantics of these various task graphs to label the videos and solve the problem in a supervised manner).

[3]https://www.lighttag.io/

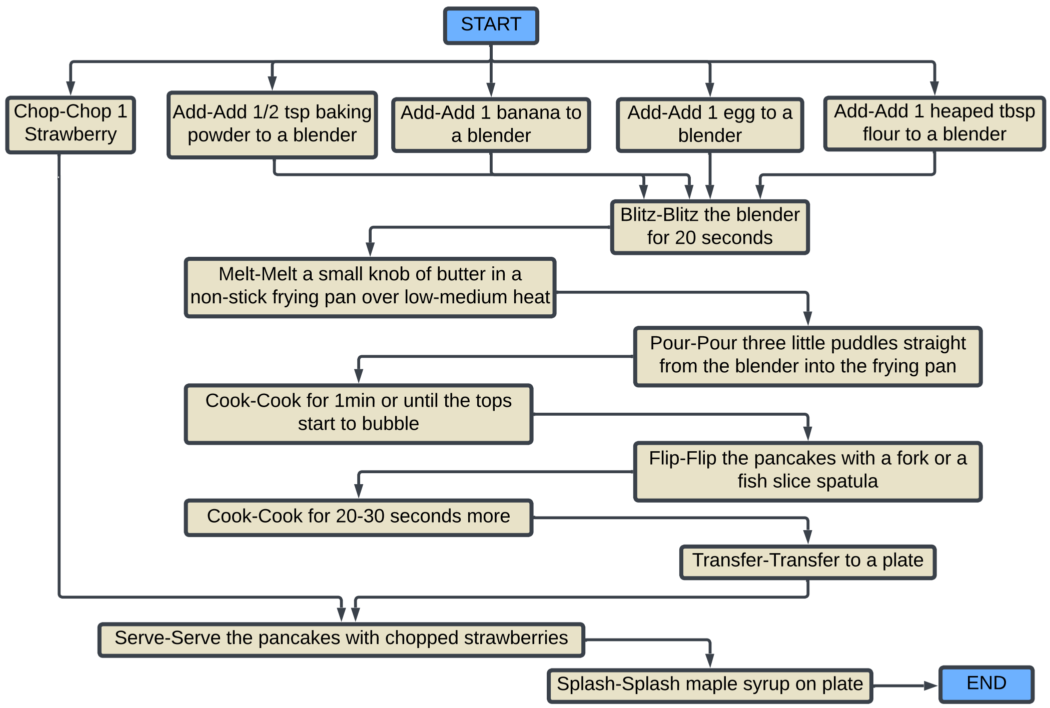

Figure 8: This graph displays the implicit dependency structure of the recipe **Blender Banana Pancakes** where the content of each node can be interpreted as {{action}-{step}} where {action} presents the description of the necessary action to be performed, and {step} presents the description as presented in the recipe text that encompasses the action, ingredients and their quantity required for the execution of the action, necessary tools used in the execution of the action, constraints on the duration of the action, how it is performed, why it is performed and other necessary settings of the environment. e.g., {Add} - {Add one banana to a blender}; here add is the necessary action and the step: Add one banana to a blender describes the action (adding), ingredient (banana), quantity (1)

C.1.2 How to record?

Sensors. Recognizing the limitations of the Hololens2 augmented reality device in capturing data, despite its advanced technology, we decided to employ a dual-device strategy [4]. While the Hololens2 offers a wealth of data from various sensors, we faced two main challenges. First, the limited field of view of the RGB camera inhibits comprehensive data capture. Second, utilizing all the secondary sensors of the Hololens2 requires operating in research mode, which, unfortunately, leads to a significant frame rate reduction for other sensory data, such as depth and monochrome cameras, when we increase the quality of the captured RGB frames.

To address these issues, we integrated a GoPro into our data-capturing system. Positioned above the Hololens2 on a head mount worn by the participant, the GoPro records 4K videos at 30 frames per second, offering a wider field of view compared to that of the Hololens2's RGB frames. This setup provides us with a more detailed perspective on the participant's activities. We use the Hololens2 in research mode to obtain a diverse range of data, including depth streams and spatial information such as head and hand movements. Additionally, we collect data from three Inertial Measurement Unit (IMU) sensors: an accelerometer, a gyroscope, and a magnetometer. This combined approach enables us to capture complete, high-quality activity data.

[4]Although we use a dual-device strategy to record activities, it's important to note that these devices aren't synchronized prior to the start of the recording process. Instead, captured footage from both devices is programmatically synchronized during post-processing using the associated timestamps

Data Capturing System. We have designed a versatile and expandable system for recording procedural activities, which can be readily adapted to meet various needs. This system has two distinct use cases: (1) as a standalone **user application** specifically designed for procedural activities and (2) as a comprehensive, plug-and-play system that functions beyond being just a user application. In its first mode, the application serves a dual role: primarily as a display interface [5] for the procedure of the activity and secondarily as a tool for noting and updating any errors made during the execution of the activity. In its second mode, the system is equipped to **capture data streams** from various sensors and allows for easy connection and configuration. This dual functionality enhances the system's adaptability, making it an efficient tool for a wide range of procedural activities.

(1) User Application. Using several illustrative snippets, we will briefly explain how our system can be used as a user interface to capture complex procedural activities, including errors. This process within our system is divided into four stages to facilitate data collection from participants:

Stage-1. First stage (Figure 9) presents the participant with a list of activities to the left. Upon selection, corresponding steps for the selected activity are then displayed on the right side of the page.

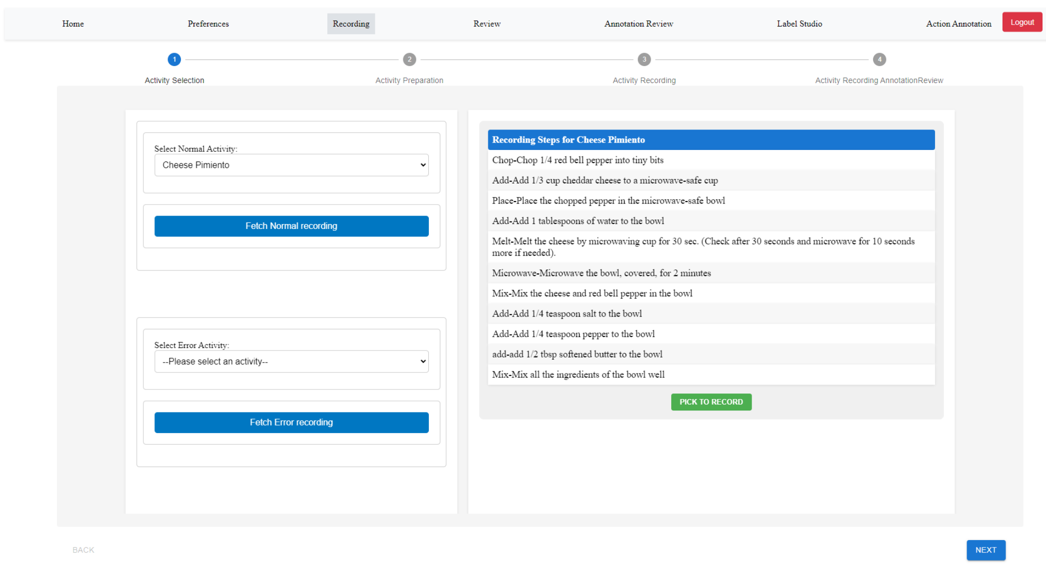

Figure 9: **Stage-1.** The participant selects the activity they wish to perform from the options presented on the left. Once a selection is made, the necessary steps for the chosen recipe will be displayed.

We provide two methods for presenting the steps of an activity, based on the input received when information about the activities is uploaded to the database:

- **Recipe Text**: If the activity's input is in plain text format, we display the text as provided.
- **Task Graph**: If the input is an action-centric task graph, we present a valid sequence of steps that conforms to the constraints defined by the graph.

Stage-2. **Activity Preparation Stage.** Although optional for a normal recording, its primary function is to prepare a script to execute during an error recording. One of our approaches to capturing error recordings involves providing participants with an interface to contemplate the errors they intend to make and modify the description of the steps for a particular activity recording session. As illustrated in Figure 10, participants can update each step based on different types of errors categorized as described above. When the participant records, they will see the updated step description as part of the sequence of steps. Moreover, GPT-4 has provided suggestions on potential errors that may occur during the activity, now available as static hint options for this recipe. However, we have observed that these *generic errors provided by GPT-4 are not particularly helpful*, as participants only considered them for script preparation in 20% of cases.

[5] Please view the tablet that displays the interface, as shown in video snippets posted on our website

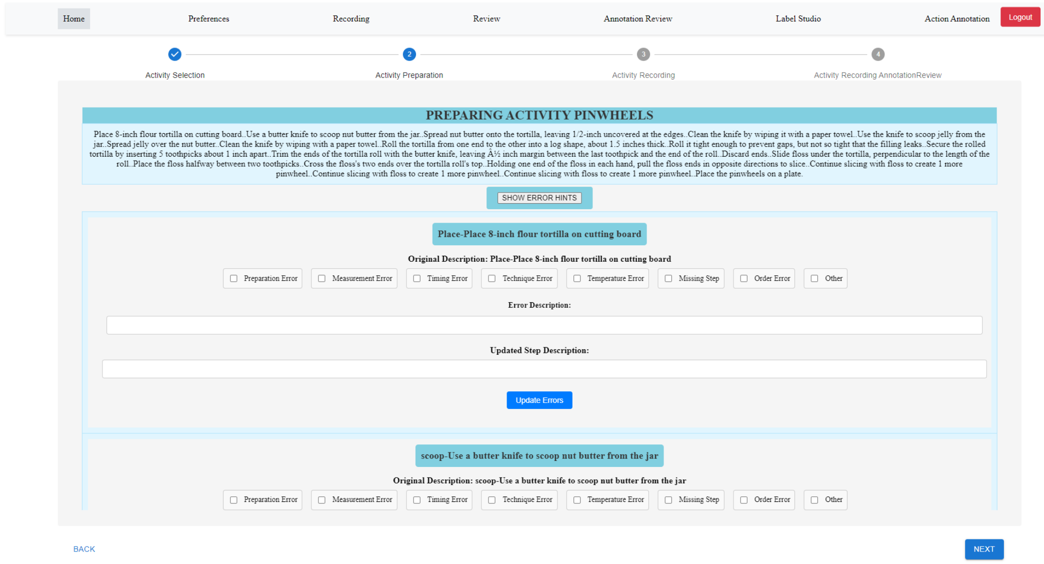

Figure 10: **Stage-2.** Interface enables participants to update step descriptions and prepare error scripts.

Stage-3. We present the participants with the sequence of steps (topological order of the task graph) for the selected activity that they will perform as illustrated in Figure 11.

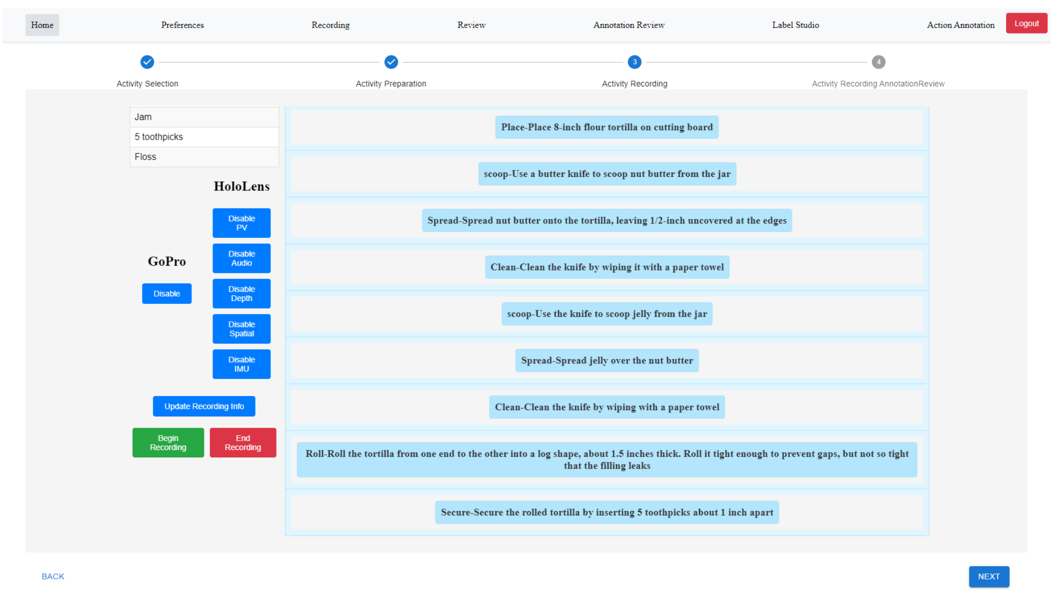

Figure 11: **Stage-3.** Displays the necessary steps to complete an activity.

Stage-4. After the data is captured either using our system or from a standalone recording system, we provide an interface to participants to review the recording they performed and correspondingly update any unplanned errors they make while performing the activity. In one of our strategies for capturing error recordings, we asked participants to induce errors *impromptu* while performing the activity. Here participants are given a series of steps corresponding to the task graph's topological order. Subsequently, participants updated information about errors they made while performing the recipe. Figure 12 presents a snippet where the participant updates one of the errors made while performing the recipe *Caprese Bruschetta*

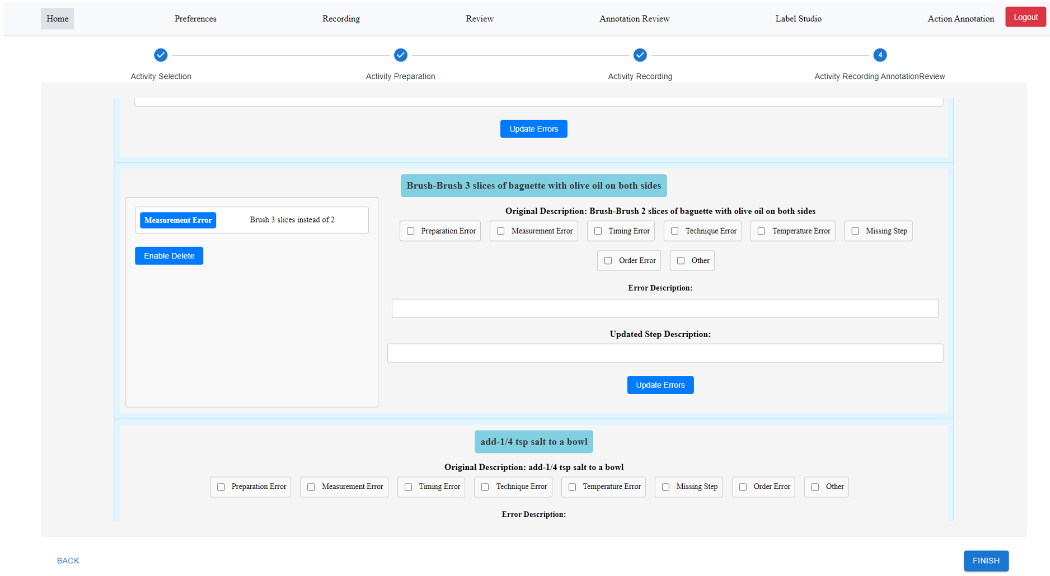

Figure 12: **Stage-4.** Similar to Stage 2, the interface allows participants to update the errors induced.

(2) Data Capturing Application. The standalone application discussed earlier can be converted into a data-capturing application by integrating several plug-and-play modules. We constructed both user and data capturing applications using the software components illustrated in Figure 13.

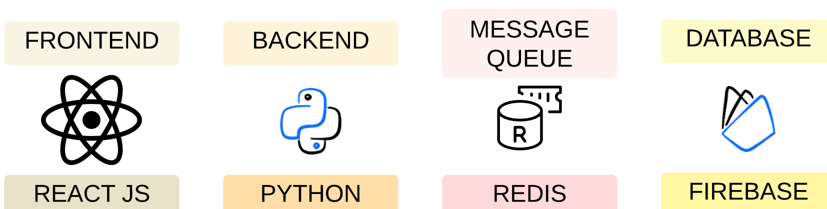

Figure 13: **Software Components** used to build the proposed system

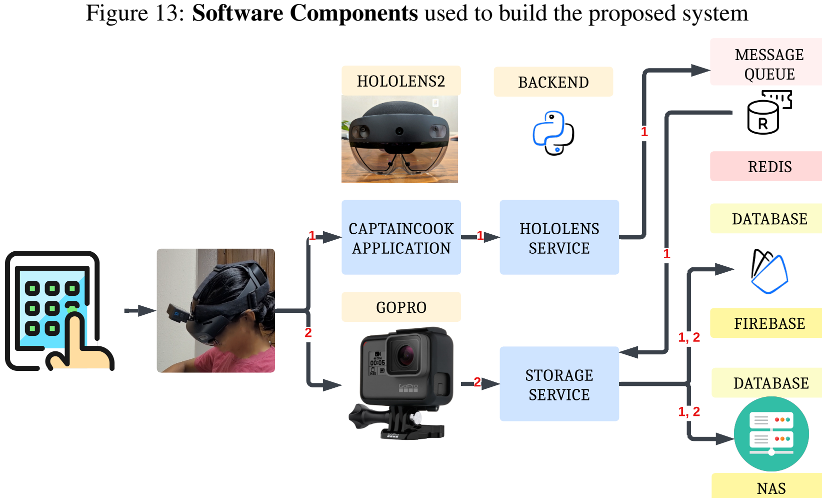

Figure 14: **Architecture** for data capturing system

In developing our data-capturing application, we have utilized data streams from various devices, specifically Hololens2 and a GoPro. The Hololens2 is particularly suited for our needs when set in research mode. It offers a wealth of data from an array of sensors, including a depth sensor, three Inertial Measurement Unit (IMU) sensors - an accelerometer, a gyroscope, and a magnetometer - and spatial information that contains head and hand tracking data. For the Hololens2, we created a custom Unity streamer application, taking inspiration from [14]. This application acts as a server, while our Python backend application assumes the role of a client. When we initiate a recording session, we establish one TCP socket connection for each sensor to capture data. As the sensor-specific data stream is received, it is immediately pushed onto the sensor-specific Redis message queue [6] Another dedicated Python backend service polls data from these message queues, processes it and subsequently stores it on a locally configured Network Attached Storage (NAS) server. When starting a recording session with GoPro, we utilize the OpenGoPro library to communicate and capture data at the established 4K resolution and 30 FPS. The recorded video is then downloaded from the GoPro through WiFi and saved onto the local NAS server. This architecture, as illustrated in Figure 14, enables us to capture, process, and securely store vast amounts of data in real-time.

C.1.3 Whom to record

Participant Statistics. The statistics concerning the participants who engaged in cooking activities are presented in Figure 15. It is important to highlight that participation in the recording process was entirely voluntary, and participants received no compensation.

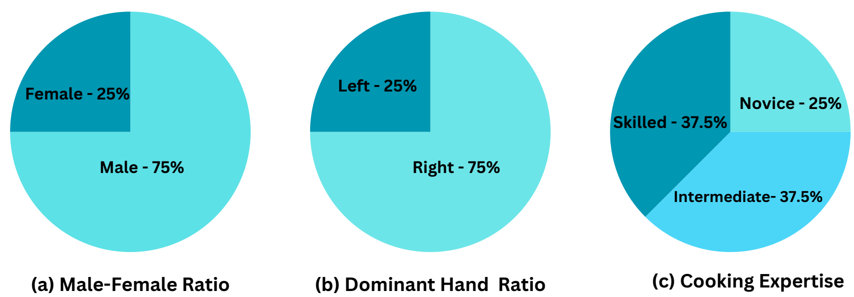

Figure 15: **Participant Statistics.** Displays information about the participants.

Participant Training. To guarantee precise data collection on cooking errors, it is essential that participants have fundamental culinary skills and thorough knowledge of the recipes they will be preparing. To assist participants, we provided them with a comprehensive list of instructional videos on basic culinary skills and techniques specific to different recipes.

Sources of bias. We recognize the inherent biases of this dataset, notably the smaller number of participants compared to traditional, large-scale datasets used for action or activity understanding. It is important to note that each participant was required to perform and record the same recipe four times. With each iteration, the recording script was altered, ensuring that each recording remained distinct. Additionally, while many errors were intentionally induced by following a script, participants also made numerous unintentional errors, which they later annotated.

[6] https://redis.com/

C.2 Data Collection

After determining what, where, and whom to record, we began collecting data from participants engaged in cooking activities. Over a period of 32 days, we conducted recordings in 10 different kitchen settings across the United States. Participants were able to schedule their availability for these activities in various kitchen environments. We provide statistics for each selected recipe, detailing the number of normal and error recordings and their respective durations in Table 11 and Figure 16.

Table 11: **Statistics.** $\mathcal{N}_n$–Count of normal recordings taken for the recipe, $\mathcal{D}_n$–Total duration of these normal recordings, $\mathcal{N}_e$–Count of error recordings taken for the recipe, $\mathcal{D}_e$–Total duration of these error recordings.

Recipe	**Steps**	$\mathcal{N}_n$	$\mathcal{D}_n$ (hrs)	$\mathcal{N}_e$	$\mathcal{D}_e$ (hrs)
Pinwheels	19	4	0.72	8	1.2
Tomato Mozzarella Salad	9	11	1.31	7	0.64
Butter Corn Cup	12	6	1.62	8	1.49
Tomato Chutney	19	7	3.34	8	2.01
Scrambled Eggs	23	6	2.69	10	3.13
Cucumber Raita	11	12	2.9	8	1.36
Zoodles	13	5	1.35	10	2.19
Microwave Egg Sandwich	12	6	1.05	12	1.67
Sauted Mushrooms	18	6	2.73	8	2.21
Blender Banana Pancakes	14	7	1.78	12	2.57
Herb Omelet with Fried Tomatoes	15	6	1.73	11	2.14
Broccoli Stir Fry	25	11	5.74	5	1.68
Pan Fried Tofu	19	8	3.38	7	2.31
Mug Cake	20	7	2.44	10	2.32
Cheese Pimiento	11	6	1.47	9	1.72
Spicy Tuna Avocado Wraps	17	7	2.0	11	2.66
Caprese Bruschetta	11	6	1.92	12	2.73
Dressed Up Meatballs	16	6	2.0	10	3.09
Microwave Mug Pizza	14	7	1.47	6	1.14
Ramen	15	10	2.40	7	1.45
Coffee	16	8	1.97	7	1.58
Breakfast Burritos	11	6	1.22	10	1.52
Spiced Hot Chocolate	7	6	0.82	10	1.01
Microwave French Toast	11	9	1.94	5	0.66
Total	384	173	50.05	211	44.41

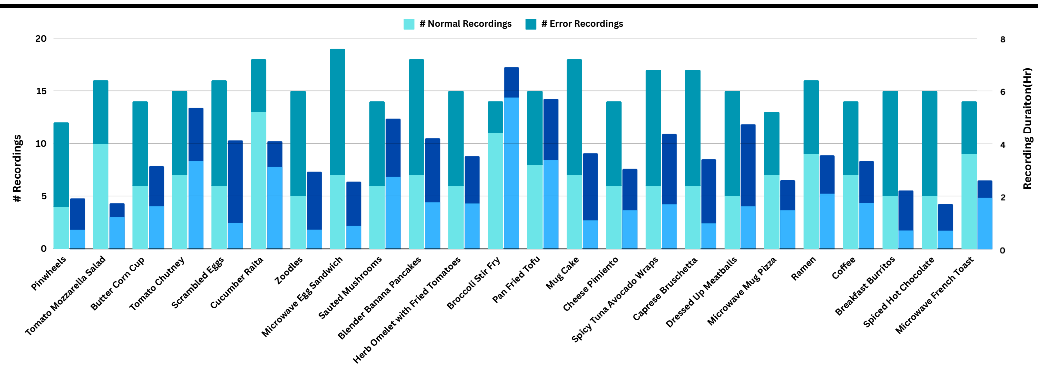

Figure 16: **Statistics.** Count and duration (in hours) statistics of normal and error recordings.

Inspection & Acclimatisation. Before initiating the recording process in each environment, participants followed a series of preparatory steps. Initially, they were instructed to remove any identifiable information from body parts visible during the recording. Additionally, they were checked to ensure they were not carrying personal identification devices, such as smartwatches containing personal data. As participants were operating in unfamiliar kitchen settings, they received a comprehensive orientation on the locations of all essential ingredients needed to complete the recipe.

Normal Recordings. Participants were provided with a tablet to access the user application described earlier. Initially, they were instructed to perform normal activities. Upon choosing a normal activity, each participant was presented with a sequence of steps that followed the topological order of the action-centric task graph constructed for that activity. Participants were expected to adhere strictly to the sequence displayed on the tablet and avoid any deviations that could lead to errors. However, participants committed numerous unintentional errors during their first execution of any given recipe and later annotated the errors induced accordingly.

Error Recordings. We developed three strategies[7] for participants to choose from, each tailored to perform the recipe in a specific environment. After choosing the strategy, participants were given detailed instructions on how to perform the recipes. We list the strategies presented to the participants (1) **Impromptu**: Participants were asked to induce errors while performing the recipe. Following the completion of each recording, participants used a web-based interface to update the errors they performed during each step. Due to the complex nature of cooking activities and the lack of experience of the participants in cooking, many errors induced in this strategy were **unintentional**. (2) **Disordered Steps**: Participants were given pre-prepared error scripts with missing steps and ordering errors. (3) **Induct Error**: Participants used a web-based interface to create an error script for each selected recipe recording. The modified recipe steps were displayed on a tablet, enabling participants to perform according to their scripted errors.

Caveats. We rely on a tablet-based interface to display the sequence of steps and also capture recordings in 4K resolution. Thus, we are aware that an OCR-based system can recognize the active step information in the tablet. To address this, we made sure that the test set included videos in which participants viewed the entire recipe instruction text as a paragraph instead of a sequence of steps.

[7] The practice of using scripted videos for activity understanding [66] has inspired us to develop the strategies.

C.3 Data Processing

C.3.1 Synchronization

After recording sessions in a kitchen environment, the data is transferred to a local NAS system and a synchronization service is run to align the raw data streams captured by the Hololens2. This includes synchronizing data from multiple streams—RGB, depth, spatial, and three Inertial Measurement Unit (IMU) sensors—using timestamps provided by the Hololens2. Post synchronization, both the raw and synchronized data are uploaded to cloud storage, and the links are made public.

C.3.2 Annotation

Coarse-Grained Action/Step Annotations. We developed an interface for performing step annotations in Label Studio[8]. This interface is used by each annotator to mark the start and end times of each step. Our steps are significantly longer than individual fine-grained actions and encompass multiple fine-grained actions required to perform the described step. Table 12 presents a summary and comparison of coarse-grained action/step annotations for our dataset alongside other popular datasets. To facilitate these annotations, we used both our user application and Label Studio. We integrated our application with Label Studio through its provided APIs, enabling the seamless creation of a labelling environment for each recording and ensuring that annotations are reliably stored.

Table 12: Comparison of coarse-grained action or step annotations across related datasets. Here, $\mathcal{T}_{avg}$ represents the avg. duration for each video, $\mathcal{N}^{seg}$ shows the total number of segments, $\mathcal{N}^{seg}_{avg}$ reveals the avg. number of segments per video, and $\mathcal{T}^{seg}_{avg}$ shows the avg. duration for all segments.

Dataset	$\mathcal{T}_{avg}$ **(min)**	$\mathcal{N}^{seg}$	$\mathcal{N}^{seg}_{avg}$	$\mathcal{T}^{seg}_{avg}$ **(sec)**
50Salads	6.4	899	18	36.8
Breakfast	2.3	11,300	6.6	15.1
Assembly 101	7.1	9523	24	16.5
CSV	0.2	18488	9.53	2.1
HoloAssist	4.48	15927	7.17	39.3
Ours (Total)	**14.8**	5300	13.8	**52.78**

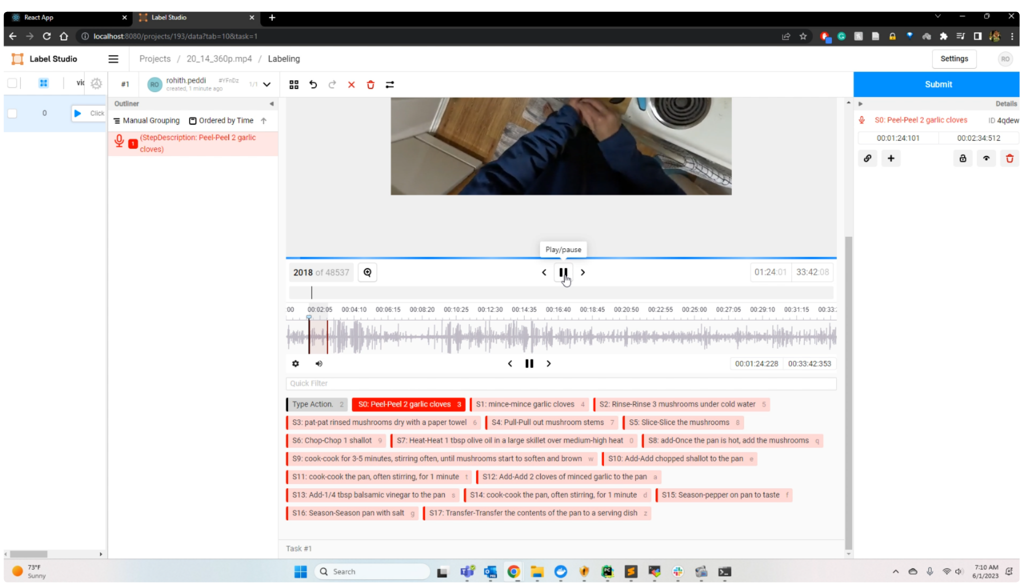

Figure 17: **Annotation Interface** developed to generate step annotations for a recording

Annotation Interface. We will briefly explain the rationale behind the design choices for our annotation interface. First, in step annotations, our goal is to define the temporal boundaries for each

[8]https://labelstud.io/

step of the recording. Consequently, we have positioned a complete list of all the steps associated with the activity beneath the video. When a time period is identified as the boundary for a specific activity step, it appears on the left-hand side of the screen. Simultaneously, the start and end times of the step are displayed on the right side in the corresponding time slots, allowing for minor adjustments.

Fine-Grained Action Annotations. Drawing inspiration from the pause-and-talk narrator [12], we have developed a web-based tool for fine-grained action annotation that leverages OpenAI's Whisper APIs for speech-to-text translation. While this system is built around the Whisper API, it is designed to be flexible enough to integrate any automatic speech recognition (ASR) system capable of handling transcription requests. Upon its acceptance, we will release this web-based annotation tool. Figure 18 illustrates the key steps for a recording and their corresponding step and action annotations.

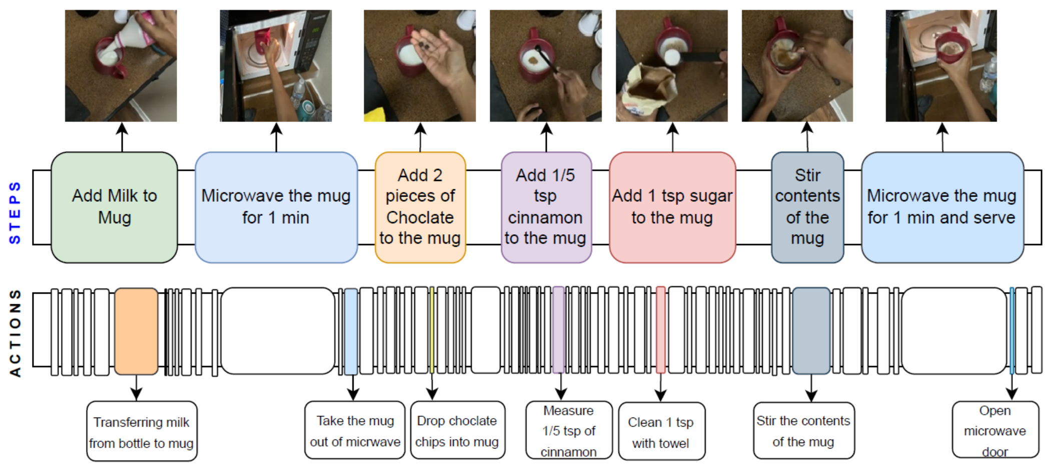

Figure 18: **Step and action annotations.** for the recipe *Spiced Hot Chocolate*.

Error annotations. Participants are required to document the errors (Appendix C.2) made during each recording. We compile the error descriptions and categorizations provided by the participants and succinctly display them, as shown in Figure 3 (see Error Categories) in the main paper. In Figure 19, we present the frequency of each error category type induced during execution.

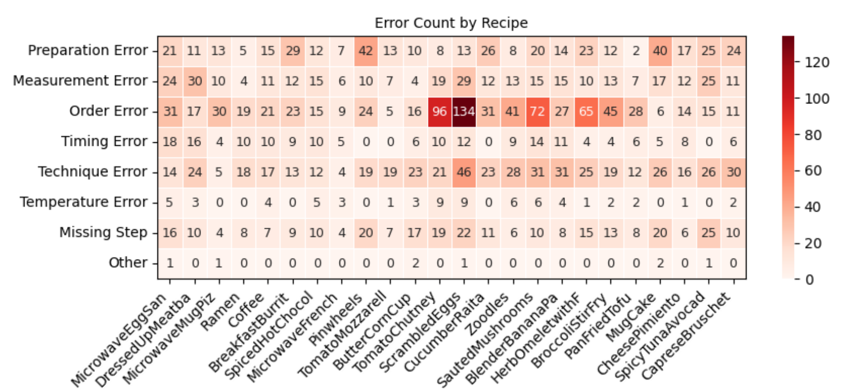

Figure 19: **Frequency of errors.** induced in the recordings for each recipe type.

C.3.3 Data Composition

In this section, we list down all the components provided as part of our data. **Raw and synchronized multi-modal data from Hololens2:** The dataset includes raw data captured using the Hololens2 device. This data is multi-modal, which means it contains information in several different forms, including visual (e.g., images or videos), auditory (e.g., sounds or speech), and others (like depth information, accelerometer readings, etc.). **4K videos from GoPro:** Includes high-resolution 4K videos recorded using a GoPro camera. Such high-resolution video can provide much detail, particularly useful for tasks like object recognition. **Step annotations** for all the data. **Fine-grained actions for 20% of the data:** Fine-grained actions might include specifics about what objects are being manipulated, exactly what movements are being made, and so on. This data could be helpful for tasks that involve understanding or predicting specific types of actions. **Extracted features using multiple backbones for different tasks:**. We provide a comprehensive overview of the components we release with the dataset in table 13

Table 13: This table presents an overview of the components we release as part of the dataset.

Hololens2	Raw	RGB RGB pose Depth Depth pose Audio Head pose Left wrist pose Right wrist pose IMU Accelerometer IMU Gyroscope IMU Magnetometer	Synchronized	RGB RGB pose Depth Depth pose Audio Head pose Left wrist pose Right wrist pose IMU Accelerometer IMU Gyroscope IMU Magnetometer
Gopro	Raw	RGB Audio		
Annotations	Step Fine-grained Action			

C.3.4 Maintenance

The dataset will be hosted on Box data storage drives and accessible via a publicly available link. The associated website will provide information about the code, dataset, and other details.

C.3.5 License

Copyright [2023] [The University of Texas at Dallas]

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

References

[1] Yazan Abu Farha, Juergen Gall, Jürgen Gall, and Juergen Gall. MS-TCN: Multi-Stage Temporal Convolutional Network for Action Segmentation. *Computer Vision and Pattern Recognition*, pages 3575–3584, June 2019. ARXIV_ID: 2006.09220 MAG ID: 2963853051 S2ID: b4673e744d0ded47fe6df3b6314f79a41359578b.

[2] Alexandros Stergiou and D. Damen. The Wisdom of Crowds: Temporal Progressive Attention for Early Action Prediction. 2022. ARXIV_ID: 2204.13340 S2ID: 95f005320424110b9c156b4739516256345b5c36.

[3] Arka Sadhu, Tanmay Gupta, Mark Yatskar, R. Nevatia, and Aniruddha Kembhavi. Visual Semantic Role Labeling for Video Understanding. *Computer Vision and Pattern Recognition*, 2021. ARXIV_ID: 2104.00990 S2ID: 588a3b90556069ec02cd40c93b8092e21d10bb1c.

[4] Bansal, Siddhant, Arora, Chetan, and Jawahar, C. V. My View is the Best View: Procedure Learning from Egocentric Videos. *European Conference on Computer Vision*, july 2022.

[5] Behrmann, Nadine, Golestaneh, S. Alireza, Kolter, Zico, Gall, Juergen, and Noroozi, Mehdi. Unified Fully and Timestamp Supervised Temporal Action Segmentation via Sequence to Sequence Translation. *European Conference on Computer Vision*, September 2022. ARXIV_ID: 2209.00638 MAG ID: 4294435452 S2ID: fbca18b2d3b2fb964fcc03f24712dd3234f30381.

[6] Piotr Bojanowski, Rémi Lajugie, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Josef Sivic. Weakly supervised action labeling in videos under ordering constraints. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 628–643, Cham, 2014. Springer International Publishing.

[7] Piotr Bojanowski, Rémi Lajugie, Edouard Grave, Francis Bach, Ivan Laptev, Jean Ponce, Cordelia Schmid, and Cordelia Schmid. Weakly-Supervised Alignment of Video With Text. *arXiv: Computer Vision and Pattern Recognition*, May 2015. ARXIV_ID: 1505.06027 MAG ID: 2950255061 S2ID: 59ac98f3910dad473e7771ac61f796a038f1708f.

[8] Bowen Shi, Wei-Ning Hsu, Kushal Lakhotia, and Abdel-rahman Mohamed. Learning Audio-Visual Speech Representation by Masked Multimodal Cluster Prediction. *International Conference on Learning Representations*, 2022. ARXIV_ID: 2201.02184 S2ID: dc9a76b7cb690e6a95f0f07bb3d4fabb7181b68d.

[9] Boyang Wan, Boyang Wan, Boyang Wan, Wenhui Jiang, Wenhui Jiang, Wenhui Jiang, Yuming Fang, Zhiyuan Luo, Guanqun Ding, Guanqun Ding, Guanqun Ding, and Guanqun Ding. Anomaly detection in video sequences: A benchmark and computational model. *Iet Image Processing*, May 2021. ARXIV_ID: 2106.08570 MAG ID: 3164956984 S2ID: d3d5836cfd6de0441f3b84cc8b5afc3407739c76.

[10] Chien-Yi Chang, De-An Huang, Yanan Sui, Li Fei-Fei, and Juan Carlos Niebles. D3TW: discriminative differentiable dynamic time warping for weakly supervised action alignment and segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 3546–3555. Computer Vision Foundation / IEEE, 2019.

[11] D. Moltisanti, Frank Keller, Hakan Bilen, and Laura Sevilla-Lara. Learning Action Changes by Measuring Verb-Adverb Textual Relationships. *arXiv.org*, 2023. ARXIV_ID: 2303.15086 S2ID: 6db9e79524529d01a38fb16bb819e8c4301896d3.

[12] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, William Price, Will Price, Will Price, and Michael Wray. The EPIC-KITCHENS Dataset: Collection, Challenges and Baselines. *arXiv: Computer Vision and Pattern Recognition*, April 2020. ARXIV_ID: 2005.00343 MAG ID: 3022491006 S2ID: 1badccbe4a3cbf8662b924a97bbeea14fe2f1ac7.

[13] Fernando De la Torre, Jessica K. Hodgins, Adam W. Bargteil, Xavier Martin, J. Robert Macey, Alex Tusell Collado, and Pep Beltran. Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. In *Tech. report CMU-RI-TR-08-22, Robotics Institute, Carnegie Mellon University*, April 2008.

[14] Dibene, Juan C. and Dunn, Enrique. HoloLens 2 Sensor Streaming. *Cornell University - arXiv*, November 2022. ARXIV_ID: 2211.02648 MAG ID: 4308505718 S2ID: b19229b4f8667dae5017cae4df5c37086332da17.

[15] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[16] Ehsan Elhamifar and Dat Huynh. Self-supervised multi-task procedure learning from instructional videos. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XVII*, volume 12362 of *Lecture Notes in Computer Science*, pages 557–573. Springer, 2020.

[17] Ehsan Elhamifar and Zwe Naing. Unsupervised procedure learning via joint dynamic summarization. *International Conference on Computer Vision (ICCV)*, 2019.

[18] 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. *Computer Vision and Pattern Recognition*, 2022.

[19] Fangqiu Yi, Hongyu Wen, and Tingting Jiang. ASFormer: Transformer for Action Segmentation. *British Machine Vision Conference*, 2021. ARXIV_ID: 2110.08568 S2ID: b4713949345551ebb6763f83004f9da68b760b6f.

[20] A. Fathi, Xiaofeng Ren, and J. M. Rehg. Learning to recognize objects in egocentric activities. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, page 3281–3288, USA, 2011. IEEE Computer Society.

[21] Daniel Fried, Daniel Fried, Jean-Baptiste Alayrac, Phil Blunsom, Chris Dyer, Chris Dyer, Stephen Clark, and Aida Nematzadeh. Learning to Segment Actions from Observation and Narration. *Annual Meeting of the Association for Computational Linguistics*, pages 2569–2588, May 2020. ARXIV_ID: 2005.03684 MAG ID: 3035218028 S2ID: 23edba4188492f39a7aefebbd7267a6a8d9ddb74.

[22] Reza Ghoddoosian, Isht Dwivedi, Nakul Agarwal, Chiho Choi, and Behzad Dariush. Weakly-supervised online action segmentation in multi-view instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13780–13790, 2023.

[23] Sanjay Haresh, Sateesh Kumar, Huseyin Coskun, Shahram N. Syed, Andrey Konin, M. Zeeshan Zia, and Quoc-Huy Tran. Learning by aligning videos in time. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5544–5554, 2021.

[24] Bradley Hayes and Brian Scassellati. Autonomously constructing hierarchical task networks for planning and human-robot collaboration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5469–5476, 2016.

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv*, December 2015.

[26] Lisa Anne Hendricks, Bryan C. Russell, Eli Shechtman, Josef Sivic, Oliver Wang, and Trevor Darrell. Localizing Moments in Video with Temporal Language. *Conference on Empirical Methods in Natural Language Processing*, pages 1380–1390, January 2018. ARXIV_ID: 1809.01337 MAG ID: 2891456603 S2ID: 0b1ff82db09672656157874718860bee942483cf.

[27] Henghui Zhao, Isma Hadji, Nikita Dvornik, K. Derpanis, R. Wildes, and A. Jepson. P3IV: Probabilistic Procedure Planning from Instructional Videos with Weak Supervision. *Computer Vision and Pattern Recognition*, 2022.

[28] Honglu Zhou, Roberto Mart'in-Mart'in, M. Kapadia, S. Savarese, and Juan Carlos Niebles. Procedure-Aware Pretraining for Instructional Video Understanding. *arXiv.org*, 2023. ARXIV_ID: 2303.18230 S2ID: a6805beab8d036362ea1719a8801631ed59916fc.

[29] De-An Huang, Li Fei-Fei, and Juan Carlos Niebles. Connectionist temporal modeling for weakly supervised action labeling. *CoRR*, abs/1607.08584, 2016.

[30] Islam, Md Mohaiminul and Bertasius, Gedas. Long Movie Clip Classification with State-Space Video Models. *European Conference on Computer Vision*, April 2022. ARXIV_ID: 2204.01692 MAG ID: 4286768340 S2ID: 9226ae23b95b3f6891461e086d910ffeb7ac448a.

[31] 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*, Oct 2019.

[32] Jiahao Zhang, A. Cherian, Yanbin Liu, Yizhak Ben-Shabat, Cristián Rodríguez, and Stephen Gould. Aligning Step-by-Step Instructional Diagrams to Video Demonstrations. *arXiv.org*, 2023. ARXIV_ID: 2303.13800 S2ID: f517037c99311519c1134e87ac9cdbb5f07e8ce0.

[33] Junnan Li, Ramprasaath R. Selvaraju, Akhilesh Deepak Gotmare, Shafiq R. Joty, Caiming Xiong, and S. Hoi. Align before Fuse: Vision and Language Representation Learning with Momentum Distillation. *Neural Information Processing Systems*, 2021. ARXIV_ID: 2107.07651 S2ID: b82c5f9efdb2ae56baa084ca41aeddd8a665c1d1.

[34] Junru Wu, Yi Liang, Feng Han, Hassan Akbari, Zhangyang Wang, and Cong Yu. Scaling Multimodal Pre-Training via Cross-Modality Gradient Harmonization. *Neural Information Processing Systems*, 2022. ARXIV_ID: 2211.02077 S2ID: a5f4018833c6327b6dcb1f4001fcbe2e76743f3b.

[35] Leslie Pack Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the Tenth International Conference on International Conference on Machine Learning*, ICML'93, page 167–173, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[36] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017.

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

[38] Hilde Kuehne, Ali Bilgin Arslan, and Thomas Serre. The Language of Actions: Recovering the Syntax and Semantics of Goal-Directed Human Activities. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 780–787, June 2014. MAG ID: 2099614498 S2ID: 185f078accb52be4faa13e4f470a9909cc6fe814.

[39] L. Logeswaran, Sungryull Sohn, Y. Jang, Moontae Lee, and Ho Hin Lee. Unsupervised Task Graph Generation from Instructional Video Transcripts. *arXiv.org*, 2023. ARXIV_ID: 2302.09173 S2ID: 8ce0d018adb51cdf49ed79d9a06082f3a0b872e5.

[40] Jun Li, Peng Lei, Peng Lei, and Sinisa Todorovic. Weakly Supervised Energy-Based Learning for Action Segmentation. *arXiv: Computer Vision and Pattern Recognition*, September 2019. ARXIV_ID: 1909.13155 MAG ID: 2977079317 S2ID: 4950245c628b041c497577516a5da71d59cdc377.

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

[42] Neelu Madan, Nicolae-Catalin Ristea, Radu Tudor Ionescu, Kamal Nasrollahi, Fahad Shahbaz Khan, Thomas B. Moeslund, and Mubarak Shah. Self-Supervised Masked Convolutional Transformer Block for Anomaly Detection, September 2022. arXiv:2209.12148 [cs].

[43] Madeline Chantry Schiappa and Y. Rawat. SVGraph: Learning Semantic Graphs from Instructional Videos. *IEEE International Conference on Multimedia Big Data*, 2022. ARXIV_ID: 2207.08001 S2ID: ecb4f7ab11c4dae31dc895d54b788946248e0862.

[44] Jonathan Malmaud, Jonathan Huang, Vivek Rathod, Nick Johnston, Andrew Rabinovich, Kevin Murphy, Kevin Murphy, Kevin Murphy, and Kevin Murphy. What's Cookin'? Interpreting Cooking Videos using Text, Speech and Vision. *North American Chapter of the Association for Computational Linguistics*, pages 143–152, January 2015. ARXIV_ID: 1503.01558 MAG ID: 2964040984 S2ID: 59ba3b1b31e2f8adb18fb886ad3fc087081c0e38.

[45] Weichao Mao, Ruta Desai, Michael Louis Iuzzolino, and Nitin Kamra. Action dynamics task graphs for learning plannable representations of procedural tasks, 2023.

[46] Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. HowTo100M: Learning a Text-Video Embedding by Watching Hundred Million Narrated Video Clips. In *ICCV*, 2019.

[47] Pedro Morgado, Nuno Vasconcelos, Ishan Misra, and Ishan Misra. Audio-Visual Instance Discrimination with Cross-Modal Agreement. *Computer Vision and Pattern Recognition*, April 2020. ARXIV_ID: 2004.12943 MAG ID: 3020933450 S2ID: 4f9a4afc0ba500d839f7ee245513af9b87add8be.

[48] Narasimhan, Medhini, Nagrani, Arsha, Sun, Chen, Rubinstein, Michael, Darrell, Trevor, Rohrbach, Anna, and Schmid, Cordelia. TL;DW? Summarizing Instructional Videos with Task Relevance & Cross-Modal Saliency. *European Conference on Computer Vision*, August 2022. ARXIV_ID: 2208.06773 MAG ID: 4292102688 S2ID: 8742e15fb4817f88fef78d9148fd83bc00a881b5.

[49] Ram Nevatia, Tao Zhao, and Somboon Hongeng. Hierarchical Language-based Representation of Events in Video Streams. *2003 Conference on Computer Vision and Pattern Recognition Workshop*, 4:39–39, June 2003. MAG ID: 2151206977 S2ID: 61d35cb5fcd823b98bae474b1ea02467c61aa3cb.

[50] Nikita Dvornik, Isma Hadji, K. Derpanis, Animesh Garg, and A. Jepson. Drop-DTW: Aligning Common Signal Between Sequences While Dropping Outliers. *Neural Information Processing Systems*, 2021. ARXIV_ID: 2108.11996 S2ID: 802495973e50115cdf10a72d466dd89a2d6ce5bc.

[51] Nikita Dvornik, Isma Hadji, Ran Zhang, K. Derpanis, Animesh Garg, R. Wildes, and A. Jepson. StepFormer: Self-supervised Step Discovery and Localization in Instructional Videos. *arXiv.org*, 2023. ARXIV_ID: 2304.13265 S2ID: 8b23d85bc8cc5602bf7114ce0e8232aee2263c2b.

[52] Dim P. Papadopoulos, Enrique Mora, Nadiia Chepurko, Kuan Wei Huang, Ferda Ofli, and Antonio Torralba. Learning program representations for food images and cooking recipes, 2022.

[53] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[54] Rohith Peddi, Saksham Singh, Saurabh, Parag Singla, and Vibhav Gogate. Towards scene graph anticipation, 2024.

[55] Mingtao Pei, Yunde Jia, and Song-Chun Zhu. Parsing video events with goal inference and intent prediction. *Vision*, pages 487–494, November 2011. MAG ID: 2045792079 S2ID: 054e8684578eb6f85cabcfb31ada42a9b7ec8fd6.

[56] A. J. Piergiovanni, Anelia Angelova, Alexander Toshev, and Michael S. Ryoo. Adversarial generative grammars for human activity prediction. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 507–523, Cham, 2020. Springer International Publishing.

[57] Po-Yao Huang, Vasu Sharma, Hu Xu, Chaitanya K. Ryali, Haoqi Fan, Yanghao Li, Shang-Wen Li, Gargi Ghosh, J. Malik, and Christoph Feichtenhofer. MAViL: Masked Audio-Video Learners. *arXiv.org*, 2022. ARXIV_ID: 2212.08071 S2ID: 8fa9db70b22b68534befe05bd75652713c49879f.

[58] Francesco Ragusa, Antonino Furnari, Salvatore Livatino, Salvatore Livatino, and Giovanni Maria Farinella. The MECCANO Dataset: Understanding Human-Object Interactions from Egocentric Videos in an Industrial-like Domain. *arXiv: Computer Vision and Pattern Recognition*, 2020.

[59] Shuhuai Ren, Linli Yao, Shicheng Li, Xu Sun, and Lu Hou. Timechat: A time-sensitive multimodal large language model for long video understanding. *ArXiv*, abs/2312.02051, 2023.

[60] Rishi Hazra. EgoTV: Egocentric Task Verification from Natural Language Task Descriptions. *arXiv.org*, 2023. ARXIV_ID: 2303.16975 S2ID: 1901745a3a592f5026abd1e9d8435019a2a25585.

[61] Nicolae-Catalin Ristea, Neelu Madan, Radu Tudor Ionescu, Kamal Nasrollahi, Fahad Shahbaz Khan, Thomas B. Moeslund, and Mubarak Shah. Self-Supervised Predictive Convolutional Attentive Block for Anomaly Detection, March 2022. arXiv:2111.09099 [cs].

[62] Sarkar, Pritam and Etemad, Ali. XKD: Cross-modal Knowledge Distillation with Domain Alignment for Video Representation Learning. *Cornell University - arXiv*, November 2022. ARXIV_ID: 2211.13929 MAG ID: 4310282783 S2ID: 8e3fb0a7cbbe27ebbab9833cff2a60b0ed61e516.

[63] Tim J. Schoonbeek, Tim Houben, Hans Onvlee, Peter H. N. de With, and Fons van der Sommen. IndustReal: A dataset for procedure step recognition handling execution errors in egocentric videos in an industrial-like setting, 2024.

[64] Luigi Seminara, Giovanni Maria Farinella, and Antonino Furnari. Differentiable task graph learning: Procedural activity representation and online mistake detection from egocentric videos, 2024.

[65] Fadime Sener and Angela Yao. Zero-shot anticipation for instructional activities. In *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*, pages 862–871. IEEE, 2019.

[66] Gunnar A. Sigurdsson, Gül Varol, X. Wang, Ali Farhadi, Ivan Laptev, and Abhinav Kumar Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. In *European Conference on Computer Vision*, 2016.

[67] Sihan Chen, Xingjian He, Longteng Guo, Xinxin Zhu, Weining Wang, Jinhui Tang, and Jing Liu. VALOR: Vision-Audio-Language Omni-Perception Pretraining Model and Dataset. *arXiv.org*, 2023. ARXIV_ID: 2304.08345 S2ID: 03755613d50e1958a97bfaad2efb976f786fbb70.

[68] Gurkirt Singh, Suman Saha, Michael Sapienza, Philip H. S. Torr, and Fabio Cuzzolin. Online real-time multiple spatiotemporal action localisation and prediction. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[69] Sixun Dong, Huazhang Hu, Dongze Lian, Weixin Luo, Yichen Qian, and Shenghua Gao. Weakly Supervised Video Representation Learning with Unaligned Text for Sequential Videos. *arXiv.org*, 2023. ARXIV_ID: 2303.12370 S2ID: bb41cacf622a165c6e30f90be41439ea0537474c.

[70] Bilge Soran, Ali Farhadi, and Linda Shapiro. Generating notifications for missing actions: Don't forget to turn the lights off! In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, page 4669–4677, USA, 2015. IEEE Computer Society.

[71] Souček, Tomáš, Alayrac, Jean-Baptiste, Miech, Antoine, Laptev, Ivan, and Sivic, Josef. Multi-Task Learning of Object State Changes from Uncurated Videos. *Cornell University - arXiv*, November 2022. ARXIV_ID: 2211.13500 MAG ID: 4310275288 S2ID: 64066a83d282c139cd418213fa561002bfa36cf1.

[72] Sebastian Stein and Stephen J. McKenna. Combining embedded accelerometers with computer vision for recognizing food preparation activities. In *UbiComp '13: Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 729–738. Association for Computing Machinery, New York, NY, USA, September 2013.

[73] Sun, Yuchong, Xue, Hongwei, Song, Ruihua, Liu, Bei, Yang, Huan, and Fu, Jianlong. Long-Form Video-Language Pre-Training with Multimodal Temporal Contrastive Learning. *Cornell University - arXiv*, October 2022. ARXIV_ID: 2210.06031 MAG ID: 4306177974 S2ID: 2adfa4e2e9e365b3dc6eca45fcec4ecbb82e1433.

[74] Yansong Tang, Dajun Ding, Dajun Ding, Dajun Ding, Yongming Rao, Yu Zheng, Danyang Zhang, Lili Zhao, Jiwen Lu, and Jie Zhou. COIN: A Large-Scale Dataset for Comprehensive Instructional Video Analysis. *Computer Vision and Pattern Recognition*, June 2019.

[75] Tengda Han, Weidi Xie, and Andrew Zisserman. Temporal Alignment Networks for Long-term Video. *Computer Vision and Pattern Recognition*, 2022. ARXIV_ID: 2204.02968 S2ID: 0e76cf252fcc119ad87d336d439e667a9200a05c.

[76] Xin Wang, Taein Kwon, Mahdi Rad, Bowen Pan, Ishani Chakraborty, Sean Andrist, Dan Bohus, Ashley Feniello, Bugra Tekin, Felipe Vieira Frujeri, Neel Joshi, and Marc Pollefeys. 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)*, pages 20270–20281, October 2023.

[77] Weizhe Liu, Bugra Tekin, Huseyin Coskun, Vibhav Vineet, P. Fua, and M. Pollefeys. Learning to Align Sequential Actions in the Wild. *Computer Vision and Pattern Recognition*, 2021.

[78] Xin Hong, Yanyan Lan, Liang Pang, J. Guo, and Xueqi Cheng. Visual Transformation Telling. *arXiv.org*, 2023. ARXIV_ID: 2305.01928 S2ID: 10ae25755d9aa17ebc9934cb4c4208c969c4e35c.

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

[80] Yichen Qian, Weixin Luo, Dongze Lian, Xu Tang, P. Zhao, and Shenghua Gao. SVIP: Sequence VerIfication for Procedures in Videos. *Computer Vision and Pattern Recognition*, 2021. ARXIV_ID: 2112.06447 S2ID: 347e9ce9465e6d9582adbb1c0658d8c26179d0bc.

[81] Yizhen Chen, Jie Wang, Lijian Lin, Zhongang Qi, Jin Ma, and Ying Shan. Tagging before Alignment: Integrating Multi-Modal Tags for Video-Text Retrieval. *arXiv.org*, 2023. ARXIV_ID: 2301.12644 S2ID: 2e9525ebe76a1d37533539ad2f560b1b453e66f6.

[82] Yue Yang, Artemis Panagopoulou, Qing Lyu, Li Zhang, Mark Yatskar, and Chris Callison-Burch. Visual Goal-Step Inference using wikiHow. *Conference on Empirical Methods in Natural Language Processing*, January 2021. ARXIV_ID: 2104.05845 MAG ID: 4206418986 S2ID: 7c9e1282124c7c161962df3b78c1c3116deffdfd.

[83] Yue Zhao, Ishan Misra, Philipp Krahenbuhl, and Rohit Girdhar. Learning Video Representations from Large Language Models. *arXiv.org*, 2022. ARXIV_ID: 2212.04501 S2ID: 933b37b21e9d61139660088adb032ff3fdf56d86.

[84] Zhao, Xueliang, Wang, Yuxuan, Tao, Chongyang, Wang, Chenshuo, and Zhao, Dongyan. Collaborative Reasoning on Multi-Modal Semantic Graphs for Video-Grounded Dialogue Generation. *Cornell University - arXiv*, October 2022. ARXIV_ID: 2210.12460 MAG ID: 4307310837 S2ID: 256fd60c692ebe12fe2bbf65d46722f511aa3117.

[85] Luowei Zhou, Chenliang Xu, and Jason J. Corso. Towards automatic learning of procedures from web instructional videos. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 7590–7598. AAAI Press, 2018.

[86] Dimitri Zhukov, Jean-Baptiste Alayrac, Ramazan Gokberk Cinbis, David F. Fouhey, Ivan Laptev, and Josef Sivic. Cross-task weakly supervised learning from instructional videos. *arXiv: Computer Vision and Pattern Recognition*, March 2019.

[87] Dimitri Zhukov, Dimitri Zhukov, Jean-Baptiste Alayrac, Ivan Laptev, and Josef Sivic. Learning Actionness via Long-Range Temporal Order Verification. *European Conference on Computer Vision*, pages 470–487, 2020. MAG ID: 3092639633 S2ID: 1bcb053622ef73ccebbfce3d7a8663b15e0c33a8.