

CAPTAINCOOK4D: A DATASET FOR UNDERSTANDING ERRORS IN PROCEDURAL ACTIVITIES

Rohith Peddi[‡], Shivvrat Arya[†], Bharath Challa[†], Likhitha Pallapothula[†],
Akshay Vyas[†], Jikai Wang[†], Qifan Zhang[†], Vasundhara Komaragiri[†],
Eric Ragan[‡], Nicholas Ruozzi[†], Yu Xiang[†], and Vibhav Gogate[†]

ABSTRACT

Following step-by-step procedures is an essential component of various activities carried out by individuals in their daily lives. These procedures serve as a guiding framework that helps to achieve goals efficiently, whether it is assembling furniture or preparing a recipe. However, the complexity and duration of procedural activities inherently increase the likelihood of making errors. Understanding such procedural activities from a sequence of frames is a challenging task that demands an accurate interpretation of visual information and the ability to reason about the structure of the activity. To this end, we collect a new egocentric 4D dataset **CaptainCook4D** comprising 384 recordings (94.5 hours) of people performing recipes in real kitchen environments. This dataset consists of two distinct types of activity: one in which participants adhere to the provided recipe instructions and another in which they deviate and induce errors. We provide 5.3K step annotations and 10K fine-grained action annotations and benchmark the dataset for the following tasks: supervised error recognition, multistep localization, and procedure learning ¹.

1 INTRODUCTION

Remember when you prepared your favorite meal after a long day and forgot to add that crucial ingredient and then lost appetite after a few bites? Such scenarios are quite common because performing long-horizon step-by-step procedural activities increases the probability of making errors. These errors can be harmless, provided they can be rectified with little consequence. Nevertheless, when the procedures in question pertain to the medical field or complex chemical experiments, the cost of errors can be substantial. Therefore, there is a pressing need to build AI systems that can guide users in performing procedural activities [10].

A key problem we need to solve in order to build such AI systems is *procedural activity understanding*, a challenging and multifaceted task that demands interpreting what is happening —specifically, determining whether the person is following the procedure correctly or making an error, anticipating what will happen, and planning the course of action to accomplish the goal. For a system to interpret what is happening, it needs to recognize and segment actions while assessing the current state of the environment [11, 13, 42, 61, 64]. To anticipate future events, the system should be able to predict actions at the beginning of the interaction or even beforehand [7, 18, 43, 51, 51, 66]. On the other hand, planning a sequence of actions requires the system to understand the possible outcomes of these interactions [4, 32, 36]. Several datasets have been introduced to facilitate understanding of procedural activity. Most of these datasets contain only normal videos of humans performing correct procedures. For an AI system to recognize errors in human procedures, datasets with error annotations are very necessary.

In this work, we present a novel dataset to assist AI systems that solve the procedural activity understanding task, focusing specifically on improving their ability to recognize and anticipate errors. We selected cooking as a domain that is sufficiently complex and encompasses different types of

*Corresponding Author

[†]The University of Texas at Dallas

[‡]The University of Florida

¹website: <https://captaincook4d.github.io/captain-cook/>

Table 1: **Ours vs Current Procedural Datasets (with and without errors)** Our dataset not only enhances the study of tasks outlined in procedural activity datasets in existing literature but also enables a systematic investigation of errors occurring during the performance of procedural activities.

Errors	Dataset Name	Domain	Ego	Depth	Recorded	Error Labels	Errors Type	Videos	Hours	Tasks
✗	YouCook2[67]	Cooking	✗	✗	✗	-	-	2000	176	89
	50 Salads [57]	Cooking	✗	✓	✓	-	-	50	4.5	2
	EGTEA Gaze+ [39]	Cooking	✓	✗	✓	-	-	86	29	7
	MPII Cooking 2 [52]	Cooking	✗	✗	✓	-	-	273	27	67
	EgoProceL [1]	Assembly	✓	✗	✓	-	-	329	62	16
	Breakfast [38]	Cooking	✗	✗	✓	-	-	1712	77	10
✓	EgoTV [49]	Simulated	✓	✗	-	✓	Intentional	7673	168	540
	Assembly101 [17]	Toy Assembly	✓	✗	✓	Partial*	Unintentional	447	53	101
	CSV [47]	Chemistry Lab	✗	✗	✓	✗	Intentional	1940	11.1	14
	HoloAssist [60]	Assembly*	✓	✓	✓	✓	Unintentional	2221	166	350
	IndustReal [53]	Toy Assembly	✓	✓	✓	✓	Int. and Unint.	84	5.8	36
	ATA [25]	Toy Assembly	✓	✗	✓	✓	Intentional	1152	24.8	3
✓	CaptainCook4D (Ours)	Cooking	✓	✓	✓	✓	Int. and Unint.	384	94.5	24

errors that are compounding in nature and completely alter the current state of the environment with no point of return. We decided to capture data from an egocentric view despite ego motions because it helps minimize occlusions more effectively than third-person view videos.

This paper makes the following **contributions**: 1) We collected an egocentric 4D dataset that features individuals following recipes in kitchen settings. Our dataset includes two distinct types of activities: one where the participants precisely follow the given recipe guidelines and another where they deviate, making errors. 2) We provide annotations for (a) Start/End times for each step of the recipe, (b) Start/End times for each fine-grained action/interaction for 20% of the collected data, and (c) Categorize and provide a detailed description of the error performed by a participant which enabled us to gather a comprehensive overview of different error types and their concise explanations. 3) We provide baselines for the following procedure understanding tasks: supervised error recognition, multi-step localization and procedure learning.

2 RELATED WORK

There has been a significant increase in procedural datasets with errors (refer to Table 1). While they all address errors during assembly and disassembly, we focus on cooking activities. In assembly tasks, the shapes and colors of the objects remain constant. However, in cooking, the shapes and colors of ingredients evolve continuously. This requires an adjustment to ongoing transformations in shape and color, making cooking a more intricate activity. Four key features distinguish our dataset: (1) **Domain** (2) **Environment**: Unlike lab environments, we collect our dataset in real-kitchen environments. (3) **Multimodal capabilities**, and (4) **Error diversity**. In this section, we elaborate on how our dataset is particularly relevant to the various tasks of interest.

Error Recognition. Given a video clip, error recognition involves identifying the errors present in the clip. This task was initially introduced as mistake detection by Assembly-101 [17] and proposed a 3-class classification of the performed procedure to classify the clip as correct, mistake, or correction. Anomaly detection, while closely related to error recognition, differentiates itself by using static cameras and backgrounds to identify unusual or abnormal behavior. Our dataset, which encompasses a variety of error types, including timing, preparation, temperature, technique, and measurement mishaps, provides researchers with a comprehensive view of error patterns in diverse situations. Cooking is a task that involves continuous changes in the shape and color of ingredients, unlike assembly tasks that usually lack variation. This unique characteristic of cooking activity makes our dataset particularly valuable for developing error recognition methods applicable to procedural tasks in the medical sector or that involve performing chemical experiments.

Temporal Action Localization (TAL) aims to identify temporal boundaries in extended videos and classify each action instance. In general, TAL methodologies fall into two categories: two-stage and single-stage approaches. The two-stage method first generates action proposals and then classifies these actions. On the contrary, the single-stage approach conducts simultaneous action localization and classification. Several datasets, such as ActivityNet [16], THUMOS14 [35], Charades [56],

MultiTHUMOS [63], AVA [28], EPIC-KITCHENS [7], and Ego4D [27], have significantly advanced the field of TAL. While our dataset may be smaller in comparison, it offers a unique feature: it includes both normal and erroneous actions. This makes it especially valuable for evaluating the robustness of TAL methods in handling actions with deviations.

Procedure Learning is a two-part process in which all video frames are first segregated into K significant steps. Then, a logical sequence of the steps necessary to complete the task is identified [2, 3, 14, 33, 54, 68]. Existing procedural activity datasets such as CrossTask [69], COIN [58] are predominantly third-person view videos. In this light, the EgoProceL dataset [1] was compiled from videos of CMU-MMAC [8], EGTEA [20], EPIC-Tents [34], MECCANO [48]. We observe that our dataset features a higher average step length, which poses a substantially more challenging problem for algorithms developed using existing egocentric procedure learning datasets.

3 DATA COLLECTION

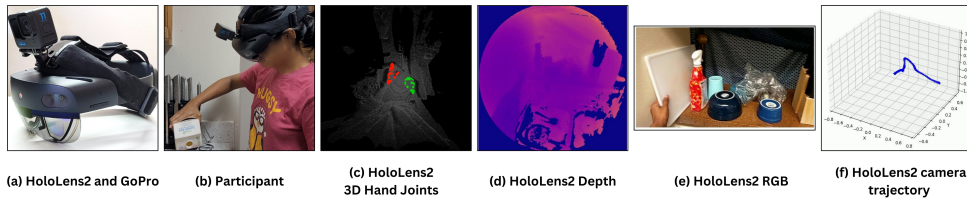


Figure 1: (a-b) display the sensor configuration for recording that includes a GoPro mounted over a HoloLens and a participant making the recipe *Cucumber Raita*, and (c-f) display the synchronized data captured by the HoloLens2 including 3D hand joints, depth, RGB and camera trajectory.

Sensors. In order to gather activity data, we employed a combination of the GoPro Hero 11 camera, which was mounted on the user’s head, and the HoloLens2 device. To facilitate data collection from HoloLens2, including its depth sensor, IMU (Inertial Measurement Unit), front RGB camera and microphone, we used a custom tool developed by [9]. Furthermore, we capture the processed head and hand tracking information provided by the HoloLens2 device. We offer data recorded from HoloLens2 and GoPro, presented separately for each recording². The GoPro provides larger field-of-view images than HoloLens2. Figure 1 illustrates the data captured from HoloLens2.

Recipes. We curated a selection of 24 cooking recipes sourced from WikiHow (Table 8), specifically focusing on recipes with a preparation time of 30 minutes or less. These recipes encompassed a wide range of culinary traditions, showcasing the diversity of cooking styles in various cuisines. Our main goal was to identify the possible errors that could occur when using different cooking tools to prepare recipes sampled from various cuisines.

Task Graphs. A task graph visually represents the sequential steps required to complete a given recipe. Each node in the task graph (for a recipe) corresponds to a step in a recipe, and a directed edge between a node x and a node y in the graph indicates that x must be performed before y . Thus, a task graph is a directed acyclic graph, and a topological sort over it represents a valid completion of the recipe. In order to construct task graphs for our collection of 24 WikiHow recipes, we meticulously identified all the essential steps involved and established their inter-dependencies, thereby establishing a topological order of tasks (see project website for details about constructed task graphs).

3.1 PROTOCOL

Our dataset was compiled by 8 participants in 10 different kitchens. Each participant selected ten recipes and recorded, on average, 48 videos in 5 different kitchens. During the filming, all participants were required to ensure that they were alone in the kitchen and remove any items that could potentially identify them, such as personal portraits, mirrors, and smartwatches with portraits. The participants used a GoPro and a HoloLens2 to record and monitor their footage. Each participant was provided with a tablet-based recording interface accessible through a web browser. To ensure

²Note that the data from GoPro and HoloLens2 are not synchronized.



Figure 2: **Error Categories:** Each row displays frames captured from different recordings of recipes, highlighting both correct and erroneous executions, with a focus on specific types of errors.

optimal video quality, we asked participants to configure the GoPro camera so that it captures videos in 4K resolution at 30 frames per second. The HoloLens2 device was programmed to stream RGB frames at a 360p resolution and a rate of 30 frames per second. It also streamed depth frames in Articulated Hand Tracking mode, referred to as “depth_ahat” mode. The device also streamed three separate IMU sensor data streams and spatial data, including head and hand poses.

3.1.1 NORMAL RECORDINGS

A recording is classified as a **normal recording** when it is captured as the participant accurately follows the procedure described in the recipe. Each participant in the study is assigned to select a recipe from the available options, which are scheduled within a kitchen setup using the recording interface. Subsequently, they are presented with one of the pre-established topological orders of the recipe, as determined by the previously constructed task graphs. The participants then proceed to follow the provided task graph, starting from the beginning and progressing through each step according to its dependencies and designated time.

3.1.2 ERROR RECORDINGS

A recording is termed an **error recording** when it is captured while the individual deviates from the recipe’s procedure, thereby inducing errors. Following the terminology used in scientific disciplines such as neuroscience [5] and chemistry, we will refer to deviations from procedures as *errors*. Note that the term “errors” used here is equivalent to what is commonly called “mistakes” in the AI community (cf. [17]). We present the count and duration statistics of recordings in Fig. 3.

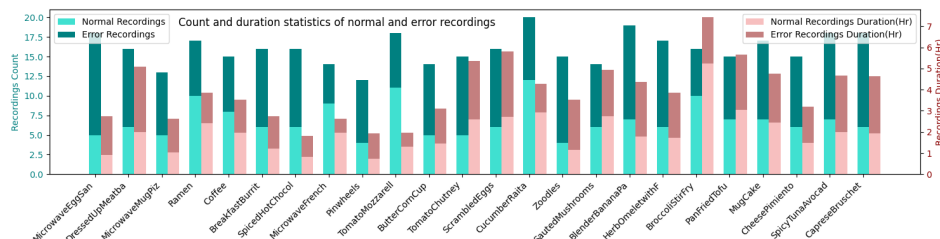


Figure 3: Count and duration (in hours) statistics of normal and error recordings.

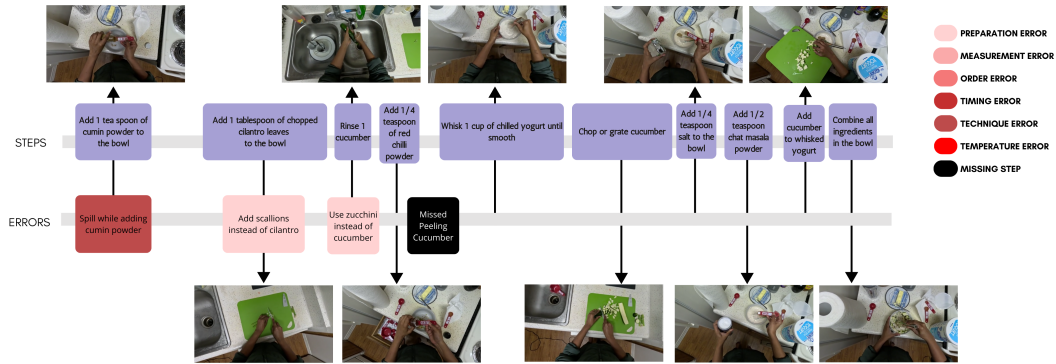


Figure 4: Displays the timeline of recipe steps and recorded errors while preparing the recipe *Cucumber Raita*. Three of four errors were intentional, but the participant unintentionally missed the *Peeling* step. In annotations, we provide step start and end times, a description, and a categorization of the error performed during that step.

Following [5, 23, 24], we classified common errors performed during a cooking activity into the following categories: (1) Preparation Error, (2) Measurement Error, (3) Technique Error, (4) Timing Error, (5) Temperature Error, (6) Missing Steps, and (7) Ordering Errors (see Figure 20). We also provide visual illustrations in Figure 2, showcasing the categorization of videos into normal and error recordings.

Error Induction. It has been a common practice to use scripted videos for activity understanding [55]. Inspired by it, we devised and implemented three strategies for participants to follow. Each participant was asked to pick a strategy for performing the recipe in a particular environment and was accordingly guided to prepare for their performance. We list the strategies presented to the participants here (1) **Pre-prepared error scripts:** In this strategy, participants were given pre-prepared error scripts with missing steps and ordering errors. (2) **Prepare error scripts:** Once the participants chose this strategy, they received a web-based interface to create an error script for each error recipe recording and displayed the modified error script on a tablet, allowing participants to perform according to their modified error scripts (3) **Impromptu:** During the later stages of the recording process, we implemented a strategy where participants were asked to induce errors while performing the recipe. Following the completion of each recording, participants were given access to a web-based interface to update errors they made during each step. Although we developed a process to capture intentional errors, due to the complex nature of cooking activities and the lack of experience of the participants in cooking, many induced errors were unintentional (Figure 4 presents one such example).

3.2 DATA ANNOTATION

Our annotations comprise (1) Annotations for coarse-grained actions or steps, providing the start and end times for each step within the recorded videos. (2) To support learning semi/weakly supervised approaches for action recognition and action anticipation, we have provided fine-grained action annotations for 20% of the recorded data. These annotations include the start and end times for each fine-grained action. (3) We have also categorized and provided error descriptions for the induced errors. These error descriptions are linked with the respective steps in the provided annotations, enabling a comprehensive understanding of the errors. Figures 4, 19 describe the granularity of different categories of annotations provided. To ensure high-quality annotations for our data, we ensured that each recording was annotated by the person who recorded the video and then reviewed by another. The reviewer was asked to double-check that all errors made by the participant in the recording were included in their corresponding step annotations.

Coarse-Grained Action/Step Annotations. We designed an interface for performing step annotations in Label Studio³. Each annotator is presented with this interface to mark the start and end times of each step. Our steps are significantly longer than a single fine-grained action and encompass multiple fine-grained actions necessary to perform the described step. For example, to accomplish

³<https://labelstud.io/>

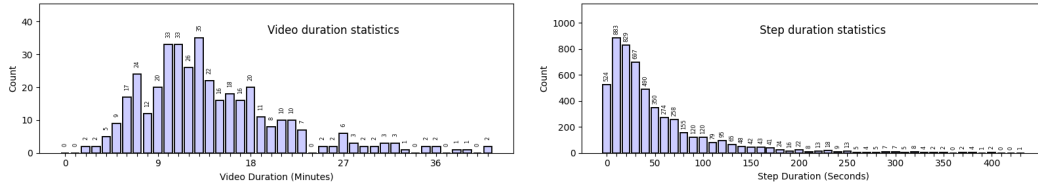


Figure 5: Displays video and step duration statistics of our dataset

the step *{Chop a tomato}*, we include the following (1) Pre-conditional actions of *{opening refrigerator, grabbing a polythene bag of tomatoes, taking a tomato, placing the tomato on cutting board, close fridge}* (2) Post-conditional actions of *{placing down the knife, grabbing the polythene bag of tomatoes, open fridge and place the bag in the fridge}*. Figure 5 presents video and step duration statistics compiled from step annotations of the dataset.

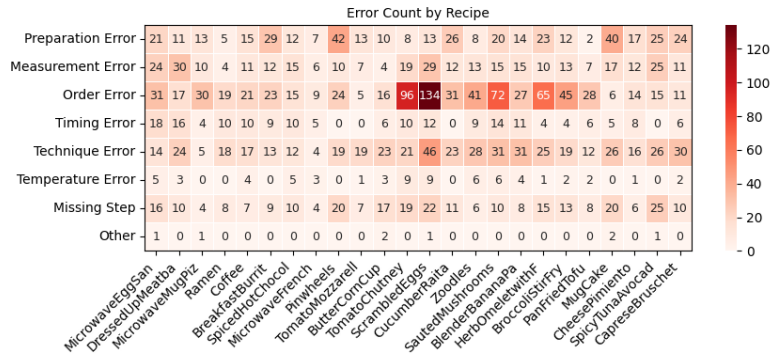


Figure 6: Displays frequency of each type of error in the recordings for a recipe.

Error Category Annotations. Following each recording, participants were also asked to categorize any errors performed in each step of the recording based on a set of guidelines. We ask participants to broadly classify an error as a (1) *Preparation Error* when they use soiled/wrong ingredients or use different tools, (2) *Measurement Error* when they use wrongly measured ingredients, (3) *Timing Error* when they perform a step in shorter or longer duration than what is prescribed (e.g. Microwave for 3 minutes instead of 30 seconds) (4) *Temperature Error* when they set higher/lower power levels in microwave or higher/lower heat levels on a stove than what is prescribed (5) *Missing Step* when they omit performing a step (6) *Technique Error* when they perform the required action incorrectly, leading to a wrong outcome than expected. (7) *Order Error* when they execute steps out of the intended sequence either intentionally or unintentionally. We compile all error categorization and description annotations and present them in Figures 6 and 20.

4 BASELINES

We provide baselines for the following tasks (1) Error Recognition, (2) Multi-Step Localization, and (3) Procedure Learning. In our approach to Error Recognition and Multi-Step Localization tasks, we utilized state-of-the-art pre-trained models originally developed for video recognition tasks to extract relevant features. Once these features were extracted, we trained distinct heads in an activity-agnostic manner, each tailored to address a specific task. We used 3D-ResNet [29], SlowFast [22], X3D [21], VideoMAE [59] and Omnivore [26] as our backbones for extracting features. We trained all our models on a single NVIDIA A40 GPU.

4.1 ERROR RECOGNITION

Utilizing error annotations provided, we propose three tasks, namely (1) Supervised Error Recognition, (2) Early Error Recognition and (3) Error Category Recognition. As baselines, we set up each task as an instance of a variant of supervised binary classification and evaluate our trained models using standard metrics for binary classification such as precision, recall, F1 score and AUC score.

4.1.1 SUPERVISED ERROR RECOGNITION

Description. In supervised error recognition, we aim to identify errors in a video segment depicting a procedural step. The presence of diverse cascading and non-cascading errors in the proposed dataset makes this setting challenging. (summarized in Figures 6 and 20). We provide a baseline, where we categorize each step into one of two classes $\{error, normal\}$ and present results in Table 2.

Table 2: Supervised Error Recognition

Baseline	Precision	Recall	F1 Score	AUC Score
3D ResNet	76.74	14.54	24.44	0.78
Slowfast	64.42	29.52	40.48	0.78
X3D	52.78	16.74	25.42	0.72
VideoMAE	75.34	25.7	38.33	0.82
Omnivore	68.24	44.49	53.87	0.84

Implementation. Firstly, We chose split by recordings as the criteria (see F.1) to construct the train, validation and test sets of recordings. Then we compiled annotated video segments corresponding to the steps of the procedural activities to prepare a comprehensive dataset of video segments, which includes 4026 training segments (with 1283 errors), 531 validation segments (179 errors), and 743 testing segments (227 errors). We utilized the error categorization labels of annotated video segments to generate binary class labels for the task.

We rely on visual cues identified by pre-trained video recognition models and use the extracted features to train models for the supervised error recognition task. Specifically, we first divided each video segment into 1-second sub-segments and extracted corresponding features utilizing the pre-trained models. Then, we used the extracted features to train a neural network with a single hidden layer with ReLU activation and a sigmoid output node. We assigned the majority class among the sub-segments to the entire segment during inference. We used the Adam optimizer with a learning rate of 0.001 to train all the classifiers. We observe that our Omnivore-based model achieves the best recall, F1 and AUC scores. We also present qualitative results of trained classifiers in figure 7.

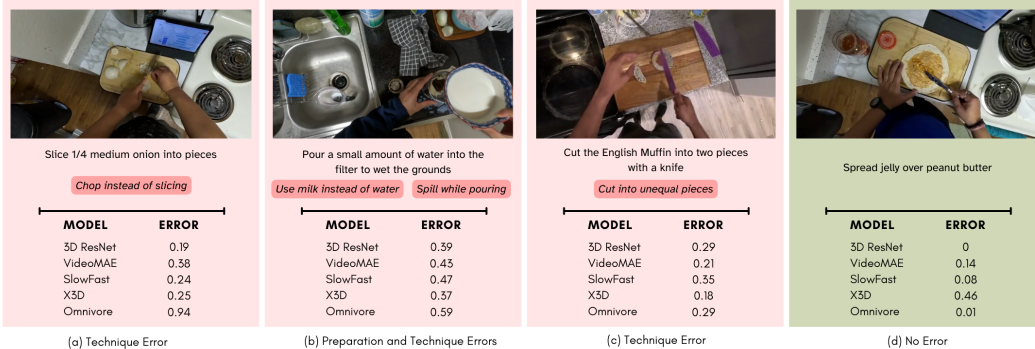


Figure 7: Displays error probabilities predicted by trained classifiers on 4 segments of the video (3 error segments and 1 normal segment) sampled from the compiled test dataset. Although our omnivore-based model outperforms the rest in classifying error segments, we note that all models are adept at distinguishing normal video segments.

Remark. The low scores indicate the complexity of the task and call for developing more sophisticated approaches. We conjecture that to improve these scores significantly, one must employ methods that seek to (semantically) understand the context, meaning, and cause of various errors⁴.

⁴We also developed methods for solving the zero-shot error recognition task (namely, training data contains only normal recordings and test data has both error and normal recordings) by adapting anomaly detection methods in the literature. However, we found that these methods perform poorly and are only slightly better than random (results are presented in the supplement). These results suggest that zero-shot error recognition is quite challenging and will require methods that seek to understand the context and meaning of errors.

4.1.2 EARLY ERROR RECOGNITION

Description. Inspired by the task of Early Action Prediction, we propose the task of Early Error Recognition. In this task, we aim to identify errors in video segments when only a partial initial segment of a step in a procedural activity is observed. We trained and evaluated models when only the initial half of the video segment was observed. We re-use the comprehensive dataset of video segments compiled in the supervised error recognition task to train a prediction head that consisted of a neural network with a single hidden layer with ReLU activation and a sigmoid output node. We present the results obtained on the evaluation of trained models in Table 3 and observe that the results are aligned with those obtained for supervised error recognition as shown in Table 2. Notice that although the AUC scores are roughly the same for both tasks, the F1 scores are always lower for early error recognition because recall is low, indicating that recognizing errors with less information is significantly more challenging.

Table 3: Early Error Recognition

Baseline	Precision	Recall	F1 Score	AUC Score
3D ResNet	72.73	3.52	6.72	0.77
Slowfast	72.41	9.25	16.41	0.75
X3D	58.33	3.08	5.86	0.73
VideoMAE	70	6.54	11.97	0.82
Omnivore	72.09	13.66	22.96	0.83

4.1.3 ERROR CATEGORY RECOGNITION

Description. Leveraging the error category annotations, we propose the task of error category recognition, where we aim to recognize and categorize errors in video segments corresponding to steps in procedural activities. We provide a baseline by setting up this task as an instance of a one-vs-all supervised binary classification.

Implementation. We re-use the comprehensive dataset of video segments compiled for error recognition to construct datasets for this task. Specifically, we construct 5 datasets by varying label assignment to video segments such that each dataset constitutes video segments belonging to a particular error category as members of the positive class and all other segments, including the ones with no errors, as members of the negative class. For each dataset, we trained a neural network with a single hidden layer and a sigmoid output node using the features extracted for the video segments and presented results in Table 4. Although the models have achieved high accuracy scores, a closer examination of the recall values reveals that they have limitations in accurately categorizing errors.

Table 4: Error Category Recognition

Method Name	Technique Error				Preparation Error				Measurement Error				Temperature Error				Timing Error			
	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1	Acc.	Prec.	Rec.	F1
3D ResNet	88.56	27.91	18.18	22.02	89.77	9.30	9.76	9.52	88.43	6.98	6.12	6.52	93.14	0.00	0.00	0.00	91.52	6.98	11.54	8.70
Slowfast	82.50	19.23	30.30	23.53	83.45	10.58	26.83	15.17	82.10	9.62	20.41	13.07	85.20	0.96	12.50	1.79	84.12	5.77	23.08	9.23
X3D	83.31	9.72	10.61	10.14	87.21	12.50	21.95	15.93	85.33	8.33	12.24	9.92	89.23	0.00	0.00	0.00	87.62	4.17	11.54	6.12
VideoMAE	84.39	19.18	22.22	20.59	86.99	13.70	27.03	18.18	84.39	8.22	12.77	10.00	88.58	1.37	12.50	2.47	87.14	6.85	19.23	10.10
Omnivore	78.20	17.57	39.39	24.30	80.22	14.19	51.22	22.22	78.33	12.16	36.73	18.27	79.81	2.03	37.50	3.85	79.00	6.08	34.62	10.34

4.2 MULTI STEP LOCALIZATION

Description. Given an untrimmed, long video that captures a procedural activity, multi-step localization aims to determine each step’s start and end frames and classify them. We have framed the supervised multi-step localization task as an instance of a supervised temporal action localization (TAL) problem. This setup is particularly challenging as our dataset encompasses both normal actions and those with deviations, termed "Technique Errors" (refer to 20), and the duration of steps in our dataset exceeds that of actions in benchmark datasets used for TAL (Table 10). We employ standard metrics used in TAL methods to evaluate trained models and present results. These metrics include temporal Intersection over Union (\mathcal{I}_t), mean Average Precision (mAP), and Recall at x ($\mathcal{R}@x$).

Implementation. Firstly, we construct three datasets with varying train, validation and test sets of recordings, using three criteria, namely recording environments (\mathcal{E}), recording persons (\mathcal{P}) and recordings (\mathcal{R}) (as described in F.1). We then extract features for each dataset and train an ActionFormer [65] head for multi-step localization. During inference, we evaluated trained models on three variants of test sets, where the first variant included the entire test set (\mathcal{T}) while the second and third

variants included filtered test sets, with one comprising only normal recordings (\mathcal{T}_n) and the other only error recordings (\mathcal{T}_e). We present the results for the first variant in Table 5 and present the results for the second and third variants in Table 6. We modified the default configuration file and set the following hyper-parameters: num_classes to 353, input_dim to 1024, max_seq_len to 4096, learning rate to 0.0001 and trained all 12 models for 16 epochs.

Table 5: Multi-step localization

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 6: Multi-step localization evaluated on test sets with only normal recordings and only error recordings.

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
3D ResNet	\mathcal{E}	\mathcal{T}_n	21.4	39.51	54.39	20.07	35.69	50.74	17.1	29.36	45.3
		\mathcal{T}_e	9.74	15.31	23.2	8.31	12.69	21.45	6.22	9.08	16.57
	\mathcal{P}	\mathcal{T}_n	19.57	35.93	49.39	18.68	33.2	47.44	15.99	27.58	43.22
		\mathcal{T}_e	13.82	27.14	39.57	12.94	23.4	37.32	10.88	19.21	33.64
	\mathcal{R}	\mathcal{T}_n	20.03	35.18	47.57	19.15	32.34	46.09	16.69	27.04	41.52
		\mathcal{T}_e	13.22	25.96	37.84	12.48	23.47	36.07	10.8	19.5	31.76
Slowfast	\mathcal{E}	\mathcal{T}_n	22.48	39.57	54.14	20.86	35.97	50.51	17.2	28.28	44.75
		\mathcal{T}_e	10.11	16.16	23.32	9	13.02	20.39	7.53	9.54	15.83
	\mathcal{P}	\mathcal{T}_n	23.12	36.55	50.45	22.09	34.09	49.11	19.24	28.93	45.12
		\mathcal{T}_e	14.78	26.68	39.97	14.14	24.73	37.71	12.56	21.76	34.37
	\mathcal{R}	\mathcal{T}_n	22.78	36.46	50.1	22.03	34.35	48.13	19.62	30.08	44.88
		\mathcal{T}_e	14.11	27.52	39.19	13.34	24.91	37.19	11.9	21.53	32.39
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

We observe that among all the feature extractors used as backbones for training the ActionFormer head, Omnivore performs much better. In Appendix F.3, we present further benchmarking results where we perform an ablation study on the performance of models trained using extracted features of varying lengths. All the models perform significantly worse on test sets constructed using only error recordings compared to the ones constructed using normal ones.

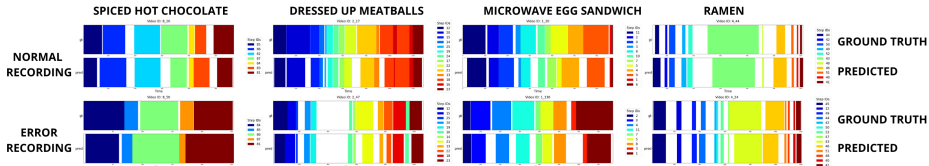


Figure 8: Qualitative results for multi-step localization.

We present qualitative results in figure 8, which displays multi-step localization results obtained when the omnivore-based model is evaluated on normal/error recordings sampled from 4 recipes namely *Spiced Hot Chocolate*, *Dressed Up Meatballs*, *Microwave Egg Sandwich* and *Ramen*.

4.3 PROCEDURE LEARNING

Description. Given long untrimmed videos of procedural activities where the sequences of steps can be performed in multiple orders, we aim to identify relevant frames across videos of activity and estimate the 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 [1, 12].

Implementation Details. We followed the setup as described in the work of [1] and trained the embedder networks for each recipe. Specifically, we train two networks, one using the Cycleback Regression loss (\mathcal{C}) proposed by [12] and the other using a blend of two loss functions: Cycleback Regression loss (\mathcal{C}) and Contrastive - Inverse Difference Moment loss (\mathcal{C}) as proposed by [1]. The combined loss function is expressed as $\mathcal{C} + \lambda \times \mathcal{C}$, where λ is a hyperparameter. (we set it to 0.5). We note that we only train embedder networks using loss functions from these methods and retain the Pro-Cut Module for assigning frames to key steps. We adhered to the hyperparameter settings specified in the original paper to train the embedder network. Utilizing an A-40 GPU, the entire training process was completed in approximately three hours for each recipe. The results are presented in Table 7; we noticed a significant decline in performance compared to the results from all other datasets reported in the paper [1]. Given that our dataset features videos with notably longer key step lengths (as indicated in Table 10), we attribute this drop in performance primarily to this distinguishing characteristic.

Table 7: **Procedure Learning.** The results showcase the performance of models trained using methods \mathcal{M}_1 [12] and \mathcal{M}_2 [1]. Where \mathcal{M}_1 employs Cycleback Regression Loss (\mathcal{C}) and \mathcal{M}_2 employs a combination of both Cycleback Regression Loss (\mathcal{C}) and Contrastive - Inverse Difference Moment Loss (\mathcal{C}). We note that we only train embedded networks using loss functions from these methods and retain the Pro-Cut Module for assigning frames to key steps. Here, \mathcal{P} represents precision, \mathcal{R} represents recall, and \mathcal{I} represents IOU.

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

5 DISCUSSION, SUMMARY AND FUTURE WORK

In this paper, we have introduced a large egocentric dataset for procedural activities. Our dataset consists of synchronized egocentric views, audio, and depth information specifically designed for tasks such as Temporal Action Segmentation, 3D activity analysis, Procedure Learning, Error Recognition, Error Anticipation, and more. Additionally, we have provided benchmarks for error recognition and Procedure Learning. While current methods have yielded promising outcomes, they continue to struggle to tackle these challenges adequately with satisfactory results, as demonstrated by our experimental assessment. This indicates the need for further exploration in this domain.

Limitations. We intend to capture deviations observed while performing a procedural activity from an egocentric view. First, we note that this type of data cannot be compiled from crowd-sourced platforms. This left us to capture participant data while performing procedural activities. Second, by the nature of the problem, errors that occur when performing procedural activities are combinatorial and can have a compounding effect. Thus, our work has the following limitations: (1) For each activity, the errors captured and presented in the dataset form a subset of the whole combinatorial space; (2) Capturing 4D data in real kitchen environments posed logistical and equipment training challenges. As a result, we were compelled to limit the data collection to a specific geographic area. (3) Compared to datasets curated from the crowd-sourced platforms used for tasks like action/activity recognition, temporal action segmentation, etc., the presented work comprises fewer recipes.

Our work opens up several avenues for future work. First, an exciting direction is the extension of the dataset to include activities from other domains. By incorporating tasks such as performing chemical experiments or executing hardware-related activities (e.g., working with cars or computer parts), the dataset can encompass a wider range of activities and provide insights into error patterns in diverse real-world scenarios. Second, the dataset can be used to compare and develop methods for solving various tasks such as transfer learning, semantic role labelling, video question answering, long video understanding, procedure planning, improving task performance, reducing errors, etc.

REFERENCES

- [1] 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.
- [2] 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.
- [3] 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.
- [4] Chien-Yi Chang, De-An Huang, Danfei Xu, Ehsan Adeli, Li Fei-Fei, Juan Carlos Niebles, Juan Carlos Niebles, and Juan Carlos Niebles. procedure planning in instructional videos. *European Conference on Computer Vision*, 2019.
- [5] Mathilde P. Chevignard, Cathy Catroppa, Jane Galvin, and Vicki Anderson. Development and evaluation of an ecological task to assess executive functioning post childhood tbi: The children’s cooking task. *Brain Impairment*, 11(2):125–143, 2010.
- [6] 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: 1badcbe4a3cbf8662b924a97bbee14fe2f1ac7.
- [7] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Antonino Furnari, Evangelos Kazakos, Jian Ma, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Rescaling Egocentric Vision: Collection, Pipeline and Challenges for EPIC-KITCHENS-100. *International Journal of Computer Vision*, October 2021.

- [8] 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-mmact) database. In *Tech. report CMU-RI-TR-08-22, Robotics Institute, Carnegie Mellon University*, April 2008.
- [9] Dibene, Juan C. and Dunn, Enrique. HoloLens 2 Sensor Streaming. *Cornell University - arXiv*, November 2022. ARXIV_ID: 2211.02648 MAG ID: 4308505718 S2ID: b19229b4f8667dae5017cae4df5c37086332da17.
- [10] Bruce Draper. DARPA’s Perceptually-enabled Task Guidance (PTG) program, 2021.
- [11] Dvornik, Nikita, Hadji, Isma, Pham, Hai, Bhatt, Dhaivat, Martinez, Brais, Fazly, Afsaneh, and Jepson, Allan D. Graph2Vid: Flow graph to Video Grounding for Weakly-supervised Multi-Step Localization. *Cornell University - arXiv*, October 2022.
- [12] Debidatta Dwivedi, 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.
- [13] Ehsan Elhamifar and Dat Huynh. Self-supervised Multi-task Procedure Learning from Instructional Videos. *European Conference on Computer Vision*, 2020.
- [14] 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.
- [15] Ehsan Elhamifar and Zhe Naing. Unsupervised procedure learning via joint dynamic summarization. *International Conference on Computer Vision (ICCV)*, 2019.
- [16] Bernard Ghanem Fabian Caba Heilbron, Victor Escorcia and Juan Carlos Nieves. ActivityNet: A large-scale video benchmark for human activity understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–970, 2015.
- [17] Fadime Sener, Dibyadip Chatterjee, Daniel Sheleпов, 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.
- [18] Fadime Sener, Rishabh Saraf, and Angela Yao. Learning Video Models from Text: Zero-Shot Anticipation for Procedural Actions. *arXiv.org*, 2021. S2ID: 3c0e77c5fb9e794336dec3872e686a91c0f653ee.
- [19] 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.
- [20] Alireza Fathi, Xiaofeng Ren, and James M. Rehg. Learning to recognize objects in egocentric activities. In *CVPR 2011*, pages 3281–3288. IEEE, June 2011.
- [21] Christoph Feichtenhofer. X3D: Expanding Architectures for Efficient Video Recognition, April 2020.
- [22] Christoph Feichtenhofer, Haoqi Fan, Jitendra Malik, and Kaiming He. SlowFast Networks for Video Recognition, October 2019.
- [23] Torun G Finnanger, Stein Andersson, Mathilde Chevignard, Gøril O Johansen, Anne E Brandt, Ruth E Hypher, Kari Risnes, Torstein B Rø, and Jan Stubberud. Assessment of executive function in everyday life-psychometric properties of the norwegian adaptation of the children’s cooking task. *Frontiers in human neuroscience*, 15:761755, 2021.
- [24] Yael Fogel, Sara Rosenblum, Renana Hirsh, Mathilde Chevignard, and Naomi Josman. Daily performance of adolescents with executive function deficits: An empirical study using a complex-cooking task. *Occupational therapy international*, 2020:3051809, 2020.
- [25] 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.

- [26] Rohit Girdhar, Mannat Singh, Nikhila Ravi, Laurens van der Maaten, Armand Joulin, and Ishan Misra. Omnivore: A Single Model for Many Visual Modalities, March 2022.
- [27] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, Miguel Martin, Tushar Nagarajan, Ilija Radosavovic, Santhosh Kumar Ramakrishnan, Fiona Ryan, Jayant Sharma, Michael Wray, Mengmeng Xu, Eric Zhongcong Xu, Chen Zhao, Siddhant Bansal, Dhruv Batra, Vincent Cartillier, Sean Crane, Tien Do, Morrie Doulaty, Akshay Erapalli, Christoph Feichtenhofer, Adriano Fragomeni, Qichen Fu, Abrham Gebreselasie, Cristina Gonzalez, James Hillis, Xuhua Huang, Yifei Huang, Wenqi Jia, Weslie Khoo, Jachym Kolar, Satwik Kottur, Anurag Kumar, Federico Landini, Chao Li, Yanghao Li, Zhenqiang Li, Karttikeya Mangalam, Raghava Modhugu, Jonathan Munro, Tullie Murrell, Takumi Nishiyasu, Will Price, Paola Ruiz Puentes, Mery Ramazanov, Leda Sari, Kiran Somasundaram, Audrey Southerland, Yusuke Sugano, Ruijie Tao, Minh Vo, Yuchen Wang, Xindi Wu, Takuma Yagi, Ziwei Zhao, Yunyi Zhu, Pablo Arbelaez, David Crandall, Dima Damen, Giovanni Maria Farinella, Christian Fuegen, Bernard Ghanem, Vamsi Krishna Ithapu, C. V. Jawahar, Hanbyul Joo, Kris Kitani, Haizhou Li, Richard Newcombe, Aude Oliva, Hyun Soo Park, James M. Rehg, Yoichi Sato, Jianbo Shi, Mike Zheng Shou, Antonio Torralba, Lorenzo Torresani, Mingfei Yan, and Jitendra Malik. Ego4D: Around the World in 3,000 Hours of Egocentric Video. *arXiv*, October 2021.
- [28] Chunhui Gu, Chen Sun, Sudheendra Vijayanarasimhan, Caroline Pantofaru, David A. Ross, George Toderici, Yeqing Li, Susanna Ricco, Rahul Sukthankar, Cordelia Schmid, and Jitendra Malik. Ava: A video dataset of spatio-temporally localized atomic visual actions. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6047–6056, 2017.
- [29] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. Learning Spatio-Temporal Features with 3D Residual Networks for Action Recognition, August 2017.
- [30] 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.
- [31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *arXiv*, December 2015.
- [32] 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.
- [33] De-An Huang, Li Fei-Fei, and Juan Carlos Nibbles. Connectionist temporal modeling for weakly supervised action labeling. *CoRR*, abs/1607.08584, 2016.
- [34] 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.
- [35] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes. <http://cvc.ucf.edu/THUMOS14/>, 2014.
- [36] Jing Bi, Jiebo Luo, and Chenliang Xu. Procedure Planning in Instructional Videos via Contextual Modeling and Model-based Policy Learning. *IEEE International Conference on Computer Vision*, 2021.
- [37] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization, January 2017.
- [38] 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.
- [39] Yin Li, Miao Liu, and James M. Rehg. In the eye of the beholder: Gaze and actions in first person video. *CoRR*, abs/2006.00626, 2020.
- [40] 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](https://arxiv.org/abs/2209.12148) [cs].

- [41] Weichao Mao, Ruta Desai, Michael Louis Iuzzolino, and Nitin Kamra. Action dynamics task graphs for learning plannable representations of procedural tasks, 2023.
- [42] Mengmeng Wang, Jiazheng Xing, and Yong Liu. ActionCLIP: A New Paradigm for Video Action Recognition. *arXiv.org*, 2021. ARXIV_ID: 2109.08472 S2ID: dc05240a06326b5b1664f7e8c95c330b08cd0349.
- [43] Antoine Miech, Ivan Laptev, Josef Sivic, Heng Wang, Lorenzo Torresani, Lorenzo Torresani, and Du Tran. Leveraging the Present to Anticipate the Future in Videos. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2019.
- [44] 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.
- [45] 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.
- [46] 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.
- [47] Yicheng Qian, Weixin Luo, Dongze Lian, Xu Tang, Peilin Zhao, and Shenghua Gao. SVIP: sequence verification for procedures in videos. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 19858–19870. IEEE, 2022.
- [48] 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.
- [49] Rishi Hazra. EgoTV: Egocentric Task Verification from Natural Language Task Descriptions. *arXiv.org*, 2023. ARXIV_ID: 2303.16975 S2ID: 1901745a3a592f5026abd1e9d8435019a2a25585.
- [50] 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].
- [51] Rohit Girdhar and Kristen Grauman. Anticipative Video Transformer. *IEEE International Conference on Computer Vision*, October 2021.
- [52] Marcus Rohrbach, Anna Rohrbach, Michaela Regneri, Sikandar Amin, Mykhaylo Andriluka, Manfred Pinkal, and Bernt Schiele. Recognizing fine-grained and composite activities using hand-centric features and script data. *International Journal of Computer Vision*, pages 1–28, 2015.
- [53] 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.
- [54] 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.
- [55] 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.
- [56] Gunnar A. Sigurdsson, Gül Varol, Xiaolong Wang, Ivan Laptev, Ali Farhadi, and Abhinav Gupta. Hollywood in homes: Crowdsourcing data collection for activity understanding. *ArXiv e-prints*, 2016.
- [57] 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.

- [58] 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.
- [59] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Neural Information Processing Systems*, 2022.
- [60] 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.
- [61] Xudong Lin, Fabio Petroni, Gedas Bertasius, Marcus Rohrbach, Shih-Fu Chang, and L. Torrè-sani. Learning To Recognize Procedural Activities with Distant Supervision. *Computer Vision and Pattern Recognition*, 2022.
- [62] 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.
- [63] Serena Yeung, Olga Russakovsky, Ning Jin, Mykhaylo Andriluka, Greg Mori, and Li Fei-Fei. Every moment counts: Dense detailed labeling of actions in complex videos. *International Journal of Computer Vision*, 2017.
- [64] Yue Yang, Joongwon Kim, Artemis Panagopoulou, Mark Yatskar, and Chris Callison-Burch. Induce, Edit, Retrieve: Language Grounded Multimodal Schema for Instructional Video Retrieval. *arXiv.org*, 2021.
- [65] Chen-Lin Zhang, Jianxin Wu, and Yin Li. Actionformer: Localizing moments of actions with transformers. In Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part IV*, volume 13664 of *Lecture Notes in Computer Science*, pages 492–510. Springer, 2022.
- [66] Zhengyuan Yang, Jingen Liu, Jing Huang, Xiaodong He, Tao Mei, Chenliang Xu, and Jiebo Luo. Cross-modal Contrastive Distillation for Instructional Activity Anticipation. *International Conference on Pattern Recognition*, 2022.
- [67] Luowei Zhou, Chenliang Xu, and Jason J. Corso. Towards Automatic Learning of Procedures from Web Instructional Videos. *arXiv*, March 2017.
- [68] 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.
- [69] 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.

APPENDICES

A Motivation for Collecting a new Dataset with Errors	17
B Data Collection—Pre-Production Phase	17
B.1 What to record?	17
B.2 How to record?	20
B.3 Whom to record	23
C Data Collection - Production Phase	25
D Data Collection - Post-Production Phase	25
D.1 Synchronization	25
D.2 Annotation	25
E Data Composition	28
F Benchmarking	29
F.1 Data Splits	29
F.2 Error Recognition	29
F.3 Multi Step Localization	33
F.4 Zero Shot Error Recognition	33

A MOTIVATION FOR COLLECTING A NEW DATASET WITH ERRORS

Current datasets that study procedural tasks, such as GTEA [19], Breakfast [38], CMU-MMAC [8], 50 Salads [57], COIN [58], CrossTask [69], ProceL [15], EgoProceL[1], Assembly101 [17], and HowTo100M [44], encompass temporal variation in the order of the steps performed. But these datasets are predominantly sourced from crowd-sourced online platforms, which results 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 the impact of errors when the objects under the interaction continuously change shape and colour during a procedural activity.

Recently, the field of error recognition in procedural activities has received significant traction, leading to the proposal of new datasets with errors [60], [25]. 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 embodying all our desired characteristics. By focusing on cooking activities which involve the desired characteristics, our dataset can be used to develop easily transferable algorithms for other domains, such as medicine and chemistry.

In subsequent sections, we will describe our data collection and annotation processes and then take a closer look at the benchmarking process. Our data collection process consists of three distinct phases: pre-production, production, and post-production.

B DATA COLLECTION—PRE-PRODUCTION PHASE

Our goal is to collect data that can help detect, segment, and analyze errors that may happen during long procedural tasks. To achieve this, we need to answer the following questions: (1) **What to record**; namely, choose the domain and tasks (recipes). (2) **How to record**; namely, choose the sensors and design a data/video capturing system. (3) **Whom to record**; namely, select participants and train them so that they can competently record the videos.

Table 8: Selected recipes categorized based on the type of required heating instrument.

Heating Instrument	Recipe	Heating Instrument	Recipe	
Kettle	Coffee	Nothing	Pinwheels	
Microwave	Breakfast Burritos	Pan	Spicy Tuna Avocado Wraps	
	Butter Corn Cup		Tomato Mozzarella Salad	
	Cheese Pimiento		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

B.1 WHAT TO RECORD?

Current popular procedural activity understanding datasets encompass recorded and curated ones from crowd-sourced online platforms. Amongst the recorded datasets, Breakfast [38], 50Salads [57], CMU-MMAC [8], and GTEA [19] capture people performing cooking activities, and Assembly-101 [17], EPIC-TENTS [34] and MECCANO [48] capture people performing activities related to assembly of toys, tents and lego blocks, respectively. Curated datasets like COIN [58], CrossTask [69], and HowTo100M [44] 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.

B.1.1 RECIPES & TASK GRAPHS

We have carefully selected 24 diverse recipes from WikiHow (see Table 8) that represent various cuisines and require the use of 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. The dummy nodes ensure that our task graphs always have one start node and one terminal node.

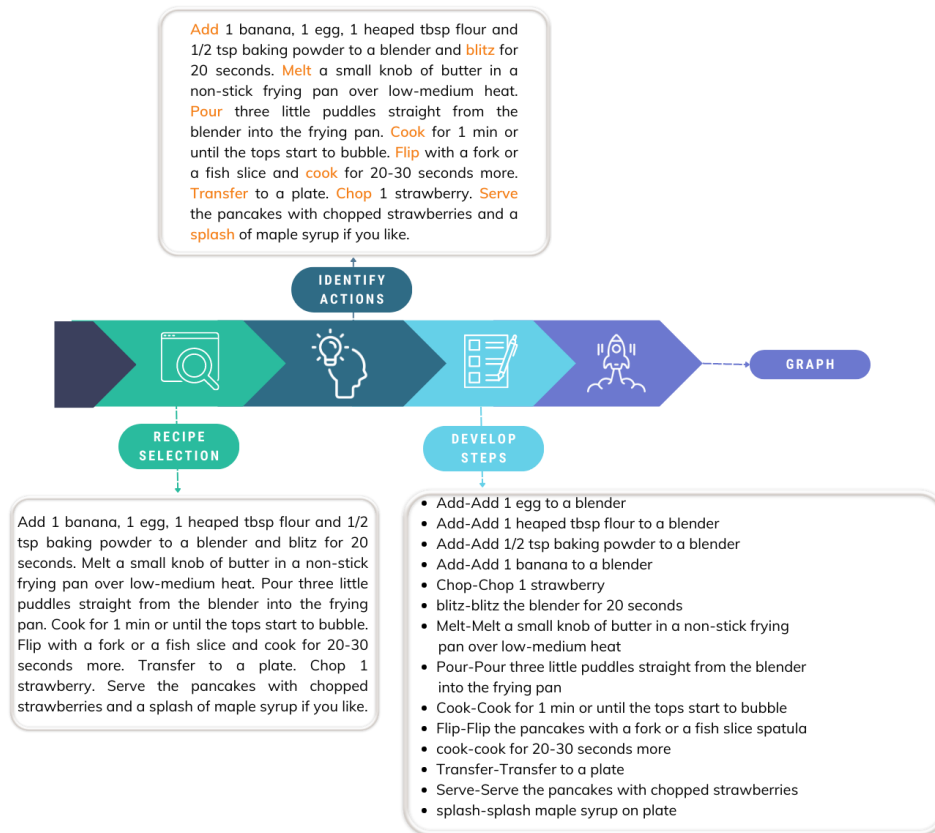


Figure 9: This figure illustrates, with an example, the three steps followed in the creation of an action-centric graph for a recipe. Given the recipe text as shown in *recipe selection*, we identify and mark all the actions necessary for the execution of the recipe as shown in *identify actions*. Once these actions are identified, we develop them into steps (as shown in *develop steps*) where each step contains only one of the actions identified before. These steps are used to construct an action-centric graph for the recipe resulting in a structure as depicted in 10

To simplify the complexity of a recipe, we have adopted a technique that uses a flow graph structure [62] to represent the dependencies between steps (think of it like a flowchart but designed for recipes). This approach helps us establish a more 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 directly impacts subsequent ones, demonstrating the dependencies between tasks.

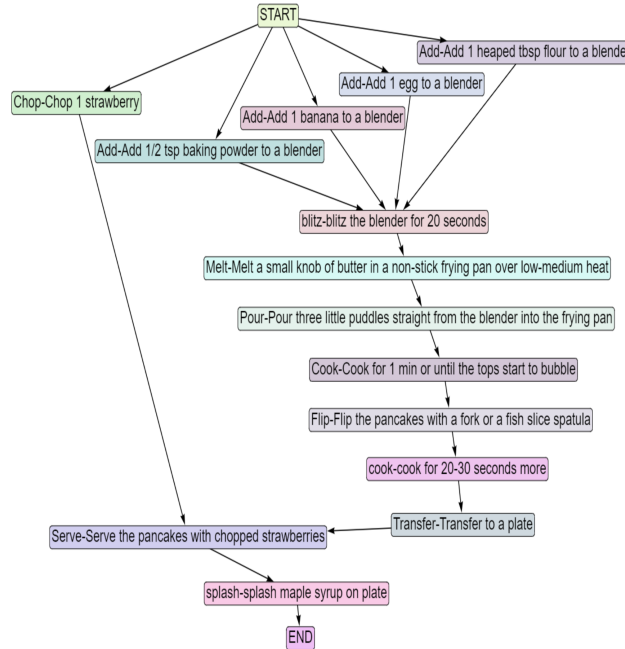


Figure 10: This graph displays the implicit dependency structure of the recipe **Blender Banana Pancakes** where the content of each node can be interpreted as $\{\{\text{action}\}-\{\text{step}\}\}$ where $\{\text{action}\}$ presents the description of the necessary action to be performed, and $\{\text{step}\}$ 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., $\{\text{Add}\} - \{\text{Add one banana to a blender}\}$; here add is the necessary action and the step: Add one banana to a blender describes the action (adding), ingredient (banana), quantity (1)

We illustrate the process we used to convert a recipe to a task graph using the recipe *Blender Banana Pancakes* (see figures 9 and 10 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 9. After we develop steps, we use a relationship annotation tool⁵ 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. *Blender Banana Pancakes*, the developed steps from 9, when represented as an action-centric graph, result in figure 10.

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 [62], Recipe Programs [45], ConjugateTaskGraphs [30], and ActionDynamicTaskGraphs [41] and our dataset can be used to learn

⁵<https://www.lighttag.io/>

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).

B.2 HOW TO RECORD?

B.2.1 SENSORS

Recognizing the limitations of the Hololens2 augmented reality device in capturing data, despite its advanced technology, we decided to employ a dual-device strategy⁶. 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.

B.2.2 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 can function in two distinct ways: (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 interface⁷ 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.

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. As depicted in Figure 11, the first stage involves presenting the participant with a list of activities on the left side of the interface. Upon selecting an activity, the corresponding steps for the selected activity are then displayed on the right side of the page.

We offer two options for presenting the steps of an activity, depending on the input provided when information about the activities is loaded into the database: (1) **Recipe Text:** If the input for an activity is plain text, we display the text exactly as provided. (2) **Task Graph:** If the input for an activity is an action-centric task graph, we provide a valid sequence of steps that adheres to the constraints defined by the graph.

Stage-2 This stage is referred to as the "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.

⁶Although 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

⁷Please view the tablet that displays the interface, as shown in video snippets posted on our website

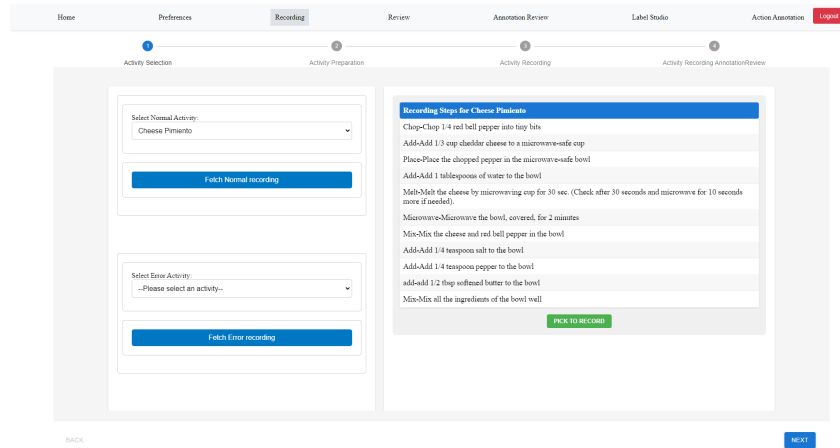


Figure 11: The interface displayed in the figure represents the initial stage of the recording cycle. Here, the participant can choose the activity they want to perform from the options presented on the left. After making a selection, the steps necessary for the chosen recipe will be shown.

As illustrated in Figure 12, 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.

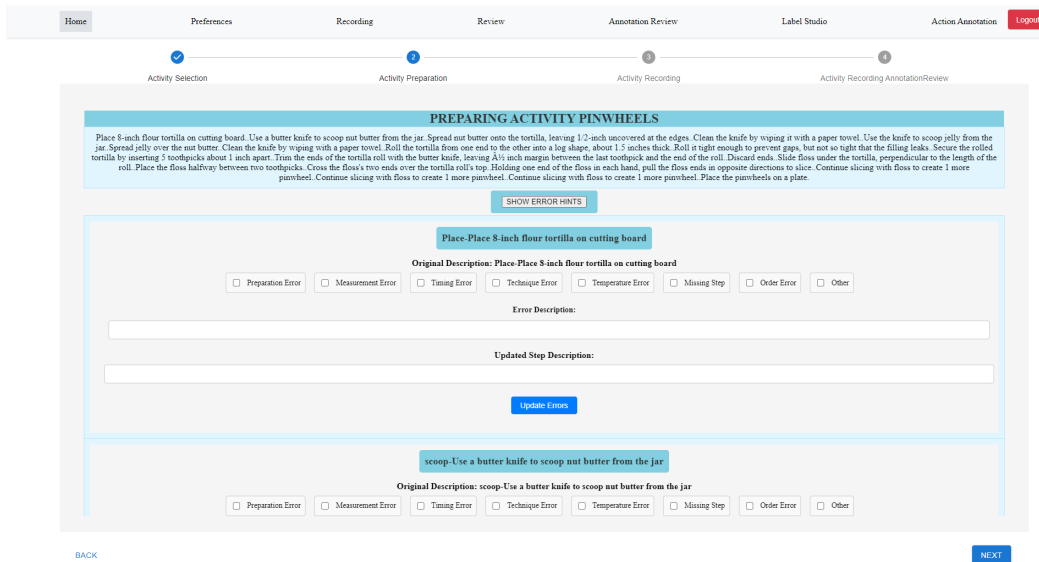


Figure 12: This interface enables participants to update step descriptions easily.

Stage-3 During this stage, we present the participants with the sequence of steps for the selected activity that they will perform. As shown in figure 13 we display each step either as plain text or present one topological order of the task graph.

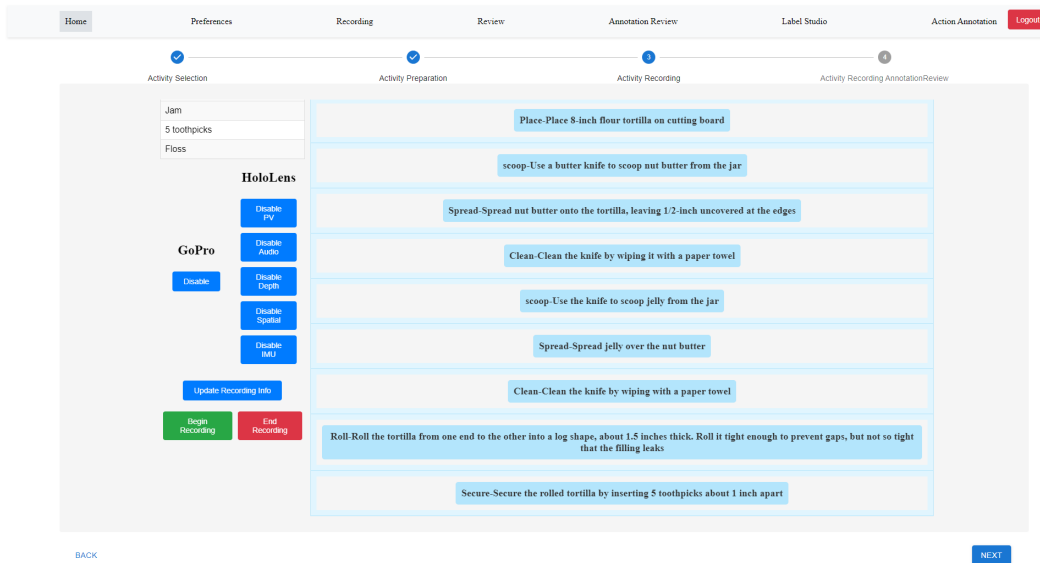


Figure 13: This interface shows the necessary steps to complete an activity.

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. In this situation, 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 14 illustrates a snippet where the participant updates one of the errors made while performing the recipe *Caprese Bruschetta*

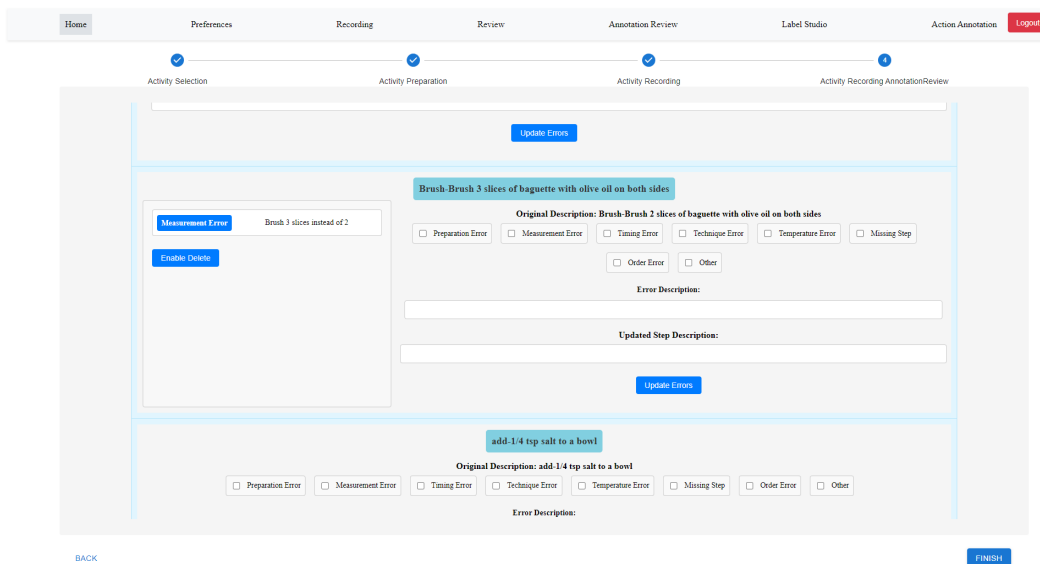


Figure 14: This interface is similar to 12, here the participant can update the information about the errors induced while performing the activity

Data Capturing Application As introduced earlier, the standalone application described above can be transformed into a data-capturing application by configuring a few plug-and-play modules. Both the user application and its extended data-capturing application are built using the software components depicted in Figure 15.

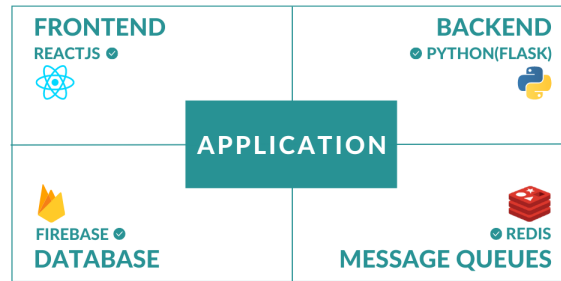


Figure 15: Software components used to build the proposed data capturing system

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 [9]. 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 queue⁸. 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 system architecture (as shown in figure 16) allows us to capture, process, and securely store vast amounts of data, all in real-time.

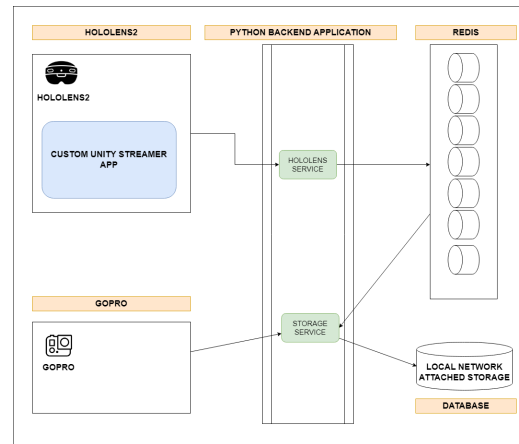


Figure 16: Figure displays the architecture for the developed data capturing system

B.3 WHOM TO RECORD

Participant Statistics. We present the statistics regarding the participants who performed cooking activities in figure 17. It is important to note that participation in the entire recording process is voluntary, and individuals were not compensated.

Participant Training. To ensure accurate data collection on cooking errors, participants must possess basic culinary skills and have complete 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.

⁸Redis (<https://redis.com/>) is an open-source, in-memory data structure store that's used to implement NoSQL key-value databases, caches, and message brokers.

Sources of bias. While this work presents the first attempt at building a comprehensive 4D dataset to study errors in procedural tasks, we acknowledge the dataset’s inherent biases. The number of participants contributing to this dataset is noticeably smaller than conventional, large-scale action or activity understanding datasets. Yet, it is important to mention that each participant is asked to perform and record the same recipe four times, and each time, the recording script changes, thus making each recording unique. Finally, note that many errors were intentional because the participants followed a script. However, they also made many unintentional errors in the process, which they annotated later.

Table 9: \mathcal{S} indicates the total number of steps in a given recipe, \mathcal{N}_n shows the count of normal recordings taken for the recipe, \mathcal{D}_n denotes the overall duration of these normal recordings, \mathcal{N}_e shows the count of error recordings taken for the recipe, \mathcal{D}_e denotes the overall duration of these error recordings

Recipe	\mathcal{S}	\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
Sauteed 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

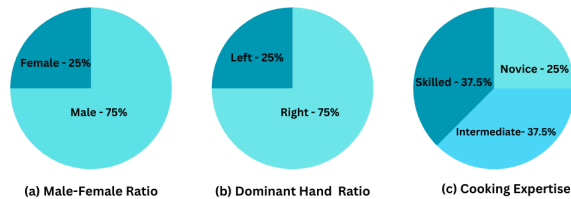


Figure 17: This data shows information about the individuals who were part of the recordings.

C DATA COLLECTION - PRODUCTION PHASE

Once we determined what, where, and whom to record, we proceeded to collect data from participants while they engaged in cooking activities. Over the course of 32 days, we recorded in 10 different kitchen settings across the United States. Participants were given the opportunity to schedule their availability to perform these activities in various kitchen environments. Table 9 describes the statistics for each selected recipe about the number of normal and error recordings and their total durations, respectively.

Inspection & Acclimatisation Before starting the recording process in every environment, participants must go through a set of steps. Firstly, they are instructed not to have any identifiable information on the body parts that will be exposed during the recording. Additionally, they are checked to ensure that they are not carrying any personal identification tools, such as a smartwatch with personal information. Since participants are filming in unfamiliar kitchen environments, they are given a detailed orientation about the location of all essential ingredients required to finish the recipe.

Data Capture - Normal Recordings Participants were given a tablet to access the user application described above. They were instructed to perform normal activities first. Upon selecting a normal activity, each participant is presented with a sequence of steps corresponding to the topological order of the constructed action-centric task graph. The participants were expected to follow the sequence of steps displayed on the tablet precisely and avoid making any errors by deviating from the given steps.

Data Capture - Error Recordings We devised and implemented three strategies for the participants to follow. Each participant was asked to choose a recording strategy for a particular environment and was guided accordingly in preparing for his recording. We list the formulated strategies here (1) **Pre-prepared error scripts**: The participants were given pre-prepared error scripts with missing steps and ordering errors. (2) **Prepare error scripts**: Once participants chose this strategy, they were given a web-based interface to create an error script for each error recipe recording and displayed the modified error script on a tablet, enabling participants to perform according to their modified error scripts (3) **Impromptu**: During the later stages of the recording process, we implemented a strategy where participants were asked to induce errors intentionally. Following the completion of each recording, participants were given access to a web-based interface to update any errors they made during each step.

Caveats Since we rely on a tablet-based interface to provide the sequence of steps, we also provide 4K videos for the recordings. We are aware that an OCR-based system might be able to recognize the content in the tablet. To tackle that, we ensured that the test set included videos where participants looked at the complete text, not just the sequence of steps.

D DATA COLLECTION - POST-PRODUCTION PHASE

D.1 SYNCHRONIZATION

Data is moved to a local Network Attached Storage (NAS) after each recording session. Upon completion of the recording cycle for each kitchen environment, a synchronization service is employed to align the raw data streams captured by the Hololens2. Data from multiple streams—including RGB, depth, spatial, and three Inertial Measurement Unit (IMU) sensors—are synchronized using timestamps provided by the Hololens2. After synchronization, both the raw and synchronized data are uploaded to the cloud.

D.2 ANNOTATION

D.2.1 COARSE-GRAINED ACTION/STEP ANNOTATIONS

We designed an interface for performing step annotations in Label Studio ⁹. Each annotator is presented with this interface to mark each step’s start and end times. Our steps are significantly

⁹<https://labelstud.io/>

longer than a single fine-grained action and encompass multiple fine-grained actions necessary for performing the described step. Table 10 summarizes and compares coarse-grained action/step annotations for our dataset as well as other popular datasets. To perform coarse-grained action/step annotations, we utilized both our user application and Label Studio. We integrated our application with Label Studio using the APIs provided by Label Studio¹⁰. This integration allowed for the seamless creation of a labelling environment for each recording and has provided a way to ensure that generated annotations are reliably stored.

Table 10: 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}_{avg}^{seg} reveals the avg. number of segments per video, and \mathcal{T}_{avg}^{seg} shows the avg. duration for all segments.

Dataset	\mathcal{T}_{avg} (min)	\mathcal{N}^{seg}	\mathcal{N}_{avg}^{seg}	\mathcal{T}_{avg}^{seg} (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

Annotation Interface. We’ll briefly explain the reasoning behind the design choices of the annotation interface. Firstly, in step annotations, we are interested in marking the temporal boundaries for each step of the recording. As a result, we’ve positioned a complete list of all steps corresponding to the activity underneath the video. After identifying a time period as the boundary for a particular activity step, this will become visible on the screen’s left-hand side. At the same time, you can see the start and end times of the step on the right side in the corresponding time slot, which can be used for minor adjustments in the created annotation.

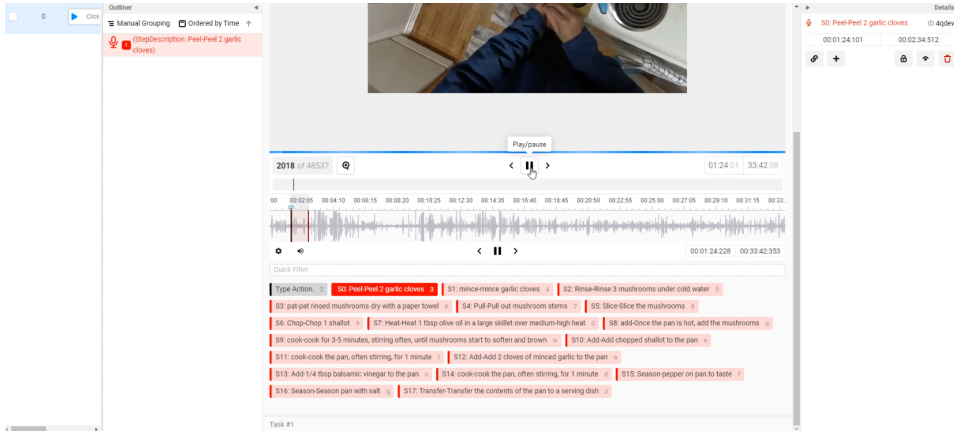


Figure 18: Annotation interface developed to generate step annotations for a recording

D.2.2 FINE-GRAINED ACTION ANNOTATIONS

Inspired by the pause-and-talk narrator [6], we have designed and developed a web-based tool for fine-grained action annotation that utilizes OpenAI’s Whisper APIs for speech-to-text translation. Even though we have built this system around the Whisper API, it is flexible enough to accommodate any automatic speech recognition (ASR) system that can serve transcription requests. Upon acceptance, we will release the developed web-based annotation tool as part of our codebase. Figure 19 illustrates the key steps for a recording and the corresponding step and action annotations.

¹⁰<https://labelstud.io/>

E 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, which is 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 11

Table 11: This table presents an overview of the components we release as part of the dataset.

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

F BENCHMARKING

In this section, we present further details on the evaluation of the curated dataset on the following tasks: (1) Supervised Error Recognition, (2) Early Error Prediction, (3) Multi-Step Localization, (4) Procedure Learning and (5) Zero-Shot Error Recognition.

F.1 DATA SPLITS

We provide a variety of splits for training models, each constructed based on specific criteria to ensure diversity in generated splits of the dataset, thus enabling models to learn to focus on different aspects of the data. These include (1) Recording Environment, (2) Recording Person, (3) Recipes (Tasks), (4) Recordings and (5) Recording Type.

Recording Environment. Our dataset includes data collected from 10 different environments. However, a larger proportion of recordings are obtained from 5 environments. We leveraged this information to create a division of the dataset, where we included the recordings from these 5 environments in both the training and validation sets while the recordings from the other environments were placed in the test set. We made sure to maintain a consistent balance of normal and error recordings across all three sets.

Recording Persons. Eight participants compiled our dataset, and each participant recorded an equal number of videos. So, we provide a split where we include recordings of two participants who performed all the recipes in the test set and the recordings by the rest of the participants in the train/validation sets.

Recipes. We carefully divided 24 chosen recipes into training, validation, and test sets, grounding our division on the specific skills each recipe demands. By identifying every skill essential for executing these recipes, we ensured that each set includes recipes that necessitate the application of these skills. This split is useful for learning tasks that require skill transfer.

Recordings. Here, we categorize all the recordings of a recipe into train, validation and test sets based on a given ratio. This split based on recordings is generated randomly and varies each time.

Recording Type. Furthermore, in tasks that require a semantic understanding of errors, methods can be employed to learn and distinguish between normal and error recordings. In such scenarios, learning using only normal recordings and identifying errors in error recordings can be incorporated.

F.2 ERROR RECOGNITION

Error recognition aims to identify and flag errors in a given video. In our specific dataset, we divided each video into segments, where each segment corresponds to a distinct step in the recipe. For benchmarking error recognition, we mark a segment as *normal* if the corresponding (recipe) step was performed correctly; otherwise, we tag it as *error*.

F.2.1 TRAINING AND EVALUATING BENCHMARK MODELS

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 error types makes solving this task particularly challenging.

Datasets and Feature Extractors. We employ features extracted by methods detailed in [21, 22, 26, 29, 59] to train our error recognition models. 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, so the last sub-segment might be shorter than 1 second. To make this last sub-segment uniform, we use zero-padding; namely, we add zeros at the end of the sub-segment to extend its duration to 1 second.

Training. At training time, we assign the class label of a segment to all its sub-segments. This yields training and validation data for learning our proposed classifiers. Specifically, in our study, we used five pre-trained models: 3D-ResNet [29], SlowFast [22], X3D [21], VideoMAE [59] and Omnivore [26], which were used to solve video recognition tasks in prior studies, and replaced their output layer by a hidden layer followed by a sigmoid node (corresponding to the class). Then, we retrained

these models using the training set and tuned the hyperparameters using the validation set. In future, we will explore other options, such as sub-sampling, to ensure fixed-sized inputs for our feature extractors.

Prediction. At prediction time, we again divide each segment into 1-second sub-segments, and after applying any necessary zero-padding, designate the class of the segment as the majority class of its sub-segments. Note that since our proposed method does not reason about order and missing step errors, we remove them from our evaluation set. In the future, we plan to use neuro-symbolic methods that leverage background knowledge about error types to improve the performance of error recognition.

Hyper-parameters. Throughout our experimental configuration, we maintained a uniform minibatch size of 512 instances. We employed ReLU activation functions in the hidden layers and sigmoid activation in the output layer. These networks were trained using the PyTorch framework [46], with the training process executed on a single NVIDIA A40 GPU. We employed the Adam optimizer [37] for training and a learning rate of 0.001. All models were trained for 100 epochs.

F.2.2 EARLY ERROR CATEGORY RECOGNITION

Table 12: Early Error Category Recognition

Type of Error	Method Name	Accuracy	Precision	Recall	F1 Score
Technique Error	3D ResNet	90.44	27.27	4.55	7.79
	Slowfast	88.83	20.69	9.09	12.63
	X3D	89.77	8.33	1.52	2.56
	VideoMAE	89.74	30	9.52	14.46
	Omnivore	87.75	20.93	13.64	16.51
Preparation Error	3D ResNet	93	0	0	0
	Slowfast	92.19	20.69	14.63	17.14
	X3D	93.41	16.67	4.88	7.55
	VideoMAE	92.34	10	5.41	7.02
	Omnivore	90.85	18.6	19.51	19.05
Measurement Error	3D ResNet	91.92	0	0	0
	Slowfast	89.77	3.45	2.04	2.56
	X3D	91.79	0	0	0
	VideoMAE	90.32	0	0	0
	Omnivore	88.16	4.65	4.08	4.35
Temperature Error	3D ResNet	94.58	0	0	0
	Slowfast	95.4	5	25	8.33
	X3D	89.94	0	0	0
	VideoMAE	94	3.57	20	6.06
	Omnivore	94.39	7.14	40	12.12
Timing Error	3D ResNet	95.29	9.09	3.85	5.41
	Slowfast	93.14	6.9	7.69	7.27
	X3D	95.15	8.33	3.85	5.26
	VideoMAE	93.64	5	3.85	4.35
	Omnivore	90.98	2.33	3.85	2.9

Table 12 presents the performance of the models for the Early Error Category Recognition task. The low recall values reveal that these models face difficulty accurately predicting errors when presented with only partial video segments.

In summary, specialized approaches are crucial for effective error recognition, especially for challenging error types such as "Temperature Error," "Measurement Error," and "Timing Error". Augmenting models with semantic information, task graphs, and other relevant attributes associated with errors is essential for achieving substantial improvements in performance.

F.2.3 FURTHER EVALUATION OF ERROR RECOGNITION AND EARLY ERROR RECOGNITION BASELINES

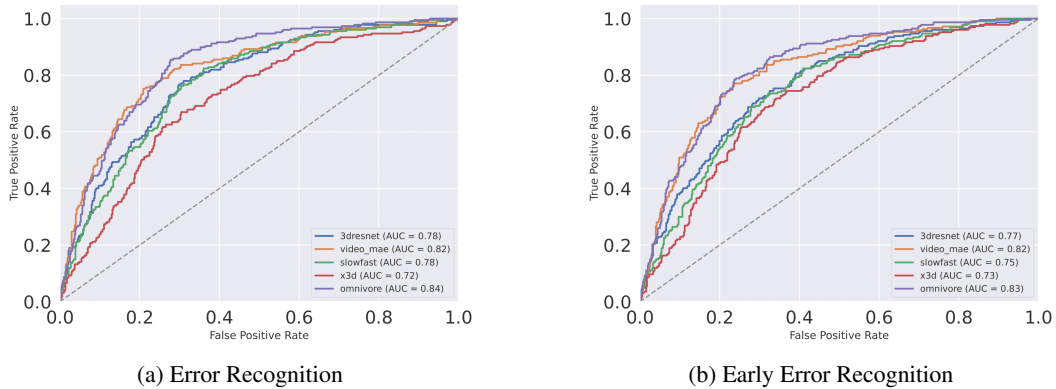


Figure 21: Evaluating Baseline Methods via ROC Curves

Figure 21a and 21b illustrate the Receiver Operating Characteristic (ROC) Curve for the baseline methods. A similar trend emerges in these visuals where Omnivore outperforms other methods while the remaining approaches exhibit comparable performance levels. Moreover, the figure underscores that video understanding methods are not optimally suited for the error recognition task. Consequently, it emphasizes the requirement for more advanced and refined methods tailored to this task.

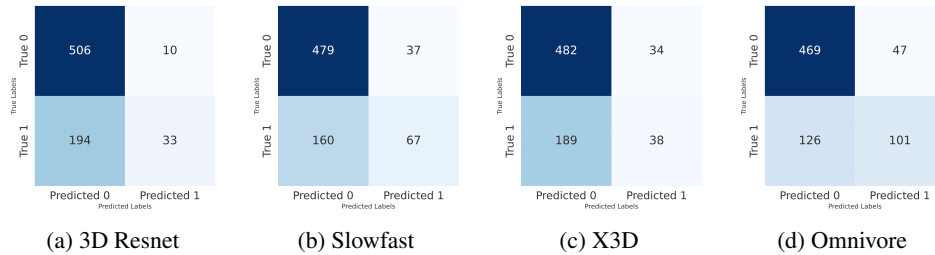


Figure 22: Confusion Matrices Demonstrating Baseline Method Performance in Supervised Error Recognition

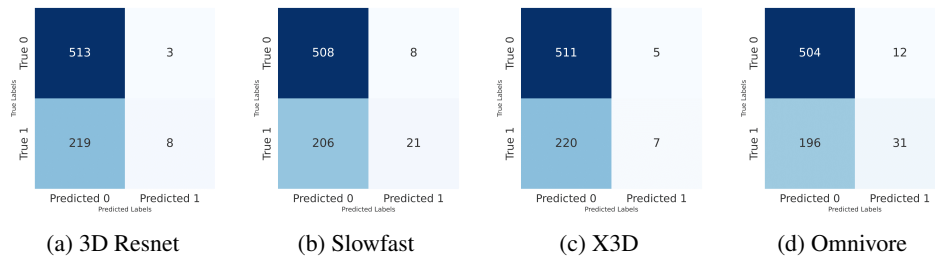


Figure 23: Confusion Matrices Demonstrating Baseline Method Performance in Early Error Detection

Figure 22 and 23 display the confusion matrices for all baseline methods in both Error Recognition and Early Error Recognition tasks. A critical observation is that all methods excel in correctly identifying non-erroneous videos, as evidenced by high values in the first row and first column. Conversely, the detection of erroneous videos remains a significant challenge for all methods, highlighting the intrinsic complexity of the task.








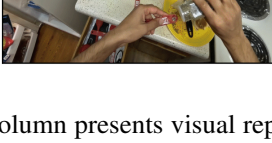
	Step Descriptions (Error)	Ground Truth	Normal
	spread-spread marinara sauce around the surface of the batter	3D Resnet	0.19
		Video MAE	0.05
		SlowFast	0.04
		X3D	0.03
		Omnivore	0.1
	spread-spread open filter in dripper to create a cone(forget to open the filter)	Ground Truth	Error
		3D Resnet	0.38
		Video MAE	0.68
		SlowFast	0.77
		X3D	0.48
	Roll-Roll the tortilla from one end to another into a log shape	Ground Truth	Normal
		3D Resnet	0.22
		Video MAE	0.09
		SlowFast	0.25
		X3D	0.23
	Roll-Roll the butter around in the mug to coat it (Roll some butter outside)	Ground Truth	Error
		3D Resnet	0.25
		Video MAE	0.13
		SlowFast	0.44
		X3D	0.29
	Mince-Mince peeled garlic cloves	Ground Truth	Normal
		3D Resnet	0.09
		Video MAE	0.02
		SlowFast	0.02
		X3D	0.29
	Cook-Cook for 2 minutes or until the zoodles are done(Spill outside while sauteing to cook)	Ground Truth	Error
		3D Resnet	0.15
		Video MAE	0.13
		SlowFast	0.73
		X3D	0.29
	add-1/4 tsp pepper to a bowl(add-1/4 tsp pepper to a plate. Unable to open the container and replaced with other pepper powder container. Spilling happens while transferring pepper to plate.)	Ground Truth	Error
		3D Resnet	0.53
		Video MAE	0.46
		SlowFast	0.56
		X3D	0.5
		Omnivore	0.64

Figure 24: The first column presents visual representations or video images. The second column offers descriptions of each step and uses red text to highlight any errors present. The third column outlines the methods evaluated. The last column displays the ground truth, indicating whether it’s an Error or Normal segment. The following rows show the predicted probabilities for the given example. A class label of 1 implies an error; Examples identified with higher probabilities are classified as errors.

F.2.4 QUALITATIVE RESULTS FOR ERROR RECOGNITION

Figure 24 presents qualitative findings across seven examples, four of which contain errors, and three are from normal segments. The second column outlines the specific steps and highlights errors in red. Among these frames, the first, third, and fifth are from normal videos, which all models successfully classified. For the frames containing errors, one example was misclassified by all baseline methods. Importantly, no error examples were correctly classified by all the methods. In one particularly challenging case, all methods failed to identify an error where butter fell outside the bowl, likely obscured by the background. These observations emphasize the inherent difficulty in error recognition and indicate that future improvements may necessitate methods designed for semantic context understanding.

F.3 MULTI STEP LOCALIZATION

The task of multi-step localization entails simultaneous recognition and localization of steps performed in a procedural activity. Here, we cast this as a Temporal Localization problem where we extract features using pretrained action recognition models and train an ActionFormer head. We further train models utilizing features extracted using Omnivore as the backbone for video segments of lengths 1sec, 3sec, and 4 seconds and present results in Table 13. We observe an improvement in the performance of the model as we increase the length of video segments used for extracting features.

Table 13: MSL using features extracted using omnivore for varying length video segments

B	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	74.9	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

F.4 ZERO SHOT ERROR RECOGNITION

Table 14: Zero Shot Error Recognition

Method	AUC	EER
SSMCTB [40]	50.65 %	49.65 %
SSPCAB [50]	50.25 %	49.74 %

To introduce an additional baseline for error recognition, we reformulate the task as a self-supervised problem focused on frame-level error recognition. More specifically, we use 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 behavior (the frame where participants made errors).

We used two self-supervised anomaly detection methods from literature, self-supervised masked convolutional transformer block (SSMCTB) [40] and self-supervised predictive convolutional attentive block (SSPCAB) [50], and trained them on top of ResNet-50 [31], where the latter serves as a neural, image-based feature extractor. Both models were trained using reconstruction loss [40]. 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 14 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 considerable difficulty of the task and underscores the necessity for more specialized approaches to achieve effective error recognition in a self-supervised manner.