EgoOops: A Dataset for Mistake Action Detection from Egocentric Videos referring to Procedural Texts

Supplementary Material

7. Details of dataset construction

7.1. Mistake definition

We define *mistakes* as "deviations from instructional steps." This definition leads to two types of mistakes: *order mistakes* and *execution mistakes*.

Order mistakes occur when there are discrepancies between the steps executed by a worker and the ones outlined in the procedural text. These include skipping necessary steps (*Missing*), swapping the step order (*Out-of-order*), pausing a step and resuming after others (*Pause-and-resume*), and inserting extra steps (*Undefined-step*).

Execution mistakes occur when a worker fails to follow the instructions while carrying out a step. We categorize them into the following six classes:

1. **Incorrect-object-use (*Object*)** executes a step with a different object specified in the text. This includes cases when the incorrect number of objects is used.
2. **Incorrect-object-picking (*Mispick*)** picks up an incorrect object, but the worker recognizes the mistake and releases the object. This mistake does not execute a step, unlike incorrect-object-use.
3. **Self-correction (*Correction*)** recognizes and corrects mistakes in a step that has been executed in a wrong way.
4. **Accidental-mistakes (*Accident*)** causes accidental happenings mainly due to carelessness.
5. **Wrong-way (*Way*)** picks a correct object but executes a step in a way that misaligns with the instruction in the text.
6. **Other-mistakes (*Others*)** induces other types of mistakes. This also includes cases where multiple types of the above mistakes occur simultaneously.

In this study, we aim to detect execution mistakes by referring to the procedure manual. However, the EgoOops dataset also includes order mistakes. These annotations are also added in the hope that they will be used in future studies.

7.2. Task selection

We aim to record procedural activities with mistake actions while following procedural texts in diverse domains. In addition, we expect to collect various categories of mistakes (*e.g.*, unintentional action, working in the wrong way). Based on these criteria, we selected the following five tasks (example frames in Fig. 7).

- **Electrical circuits** (***EC***): Connecting electrical elements to complete an electrical circuit that turns a propeller.
- **Color mixture experiments** (***CM***): Examining the color of various solutions of detergent and fluorescent paint when illuminating them with a blacklight.
- **Ionic reaction experiments** (***IR***): Examining ionic reaction by dropping chemical solutions to metal plates.
- **Toy building blocks** (***BB***): Piling up building blocks to construct the specified structure.
- **Cardboard crafts** (***CB***): Crafting Omikuji boxes, Japanese random fortunes, from cardboard.

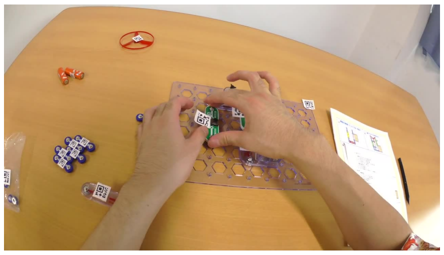
These tasks satisfy the above criteria; the domains are diverse (electronics, chemistry, assembly, and crafts) and contain various mistakes (*e.g.*, accidentally cutting off cardboard, using wrong chemicals). In addition, the workers should follow procedural texts to accomplish the task, and the video duration ranges from a few to 30 minutes.

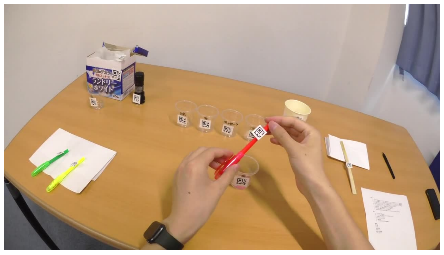
7.3. Preparation of procedural texts

One of the authors wrote a procedural text for each task before recordings. First, they prepare the draft versions referring to the following existing texts. For color mixture experiments and cardboard crafts, they searched the web to collect procedural texts. For ion reaction experiments and electrical circuits, they borrowed the texts from out-of-the-box kits. For toy building blocks, they wrote procedures from scratch because we found no resources on the web or in kits. We further revised these drafts to improve clarity (*e.g.* specify the tools to be used). These original procedural texts were written in Japanese, then we manually translated them to prepare the English versions. The English version of procedural texts is bundled into the released dataset, and participants of our recording referred to the Japanese version as explained in the next section.

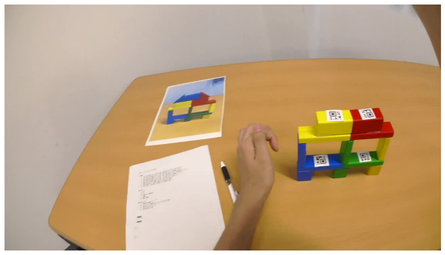
7.4. Video recording

Participants, camera, and environments. Four Japanese graduate students (4 males) performed the tasks following procedural texts. The participants were equipped with a head-mounted camera Panasonic HX-A500 as shown in Fig. 2. It is a 30 fps video camera with 4K RGB resolution. When recording the tasks, the participants referred to the Japanese versions of procedural texts. We also gave them images of the finished products in electrical circuits and toy building blocks. This is because our preliminary experiments showed that the two tasks were difficult to complete only with the texts. To avoid the influence of background changes, a desk with objects, tools, and printed procedural texts is placed in the same indoor position across the record-


(a) Electrical circuits.


(b) Color mixture experiments.

(c) Ionic reaction experiments.


(d) Toy building blocks.

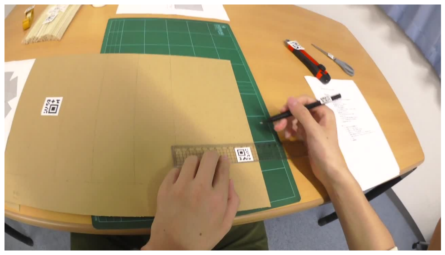

(e) Cardboard crafts.

Figure 7. Example frames of the five tasks.

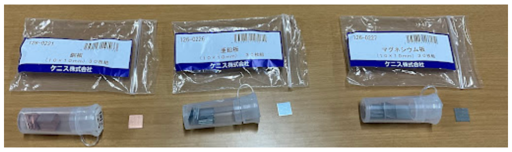


Figure 8. Raw visual cues of copper (left), zinc (center), and magnesium (right) plates. Especially, zinc and magnesium are indistinguishable.

ing sessions. The participants are recorded sitting to capture manipulated objects in detail.

Micro QR codes. Detecting objects mentioned in procedural texts from videos is crucial for identifying incorrect actions. For example, object detection can determine if a worker follows the instruction to paste papers using glue not tape. However, some objects are visually indistinguishable as shown in Fig. 8. Previous studies [16, 21, 23, 27] assumed manual annotations of the objects, yet it's costly and time-consuming. To address these issues, we attach Micro QR Codes [14] that encode the objects' names as shown in Fig. 9. This is an effortless solution because QR Code detection does not require further tuning for our dataset to apply existing detectors. Nevertheless, we cannot integrate QR Codes with our proposed approach now, thus our future work is to leverage QR Codes for object-oriented mistake detection.

Recording process. A participant worked on every task 2 or 4 times, totaling 10 recordings for each task. To record various mistake actions, they intentionally perform mistake actions that we assume in advance (*e.g.*, skipping several procedures). Out of ten times to record videos, they contained the intentional mistakes five times and tried to follow procedures correctly for the other five times. Note that accidental mistakes due to carelessness also happened even in the latter, which is desirable to contain natural mistakes in EgoOops.

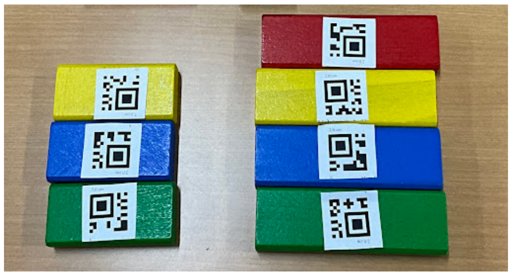


Figure 9. Micro QR codes on objects. It's difficult to specifically identify the short thin cuboids (left) and the long thin cuboids (right) only from vision. Micro QR codes are useful for distinguishing them.

7.5. Annotation guidelines

Video-text alignment. We first annotate video-text alignment by extracting start and end timestamps (i.e., segments) and mapping them to the corresponding steps. To reduce the ambiguity of segment annotations, we define the segment period as a time range from grasping the step-related objects to releasing them. We also localize extra step segments not written in a procedural text, and they are labeled as *undefined* (*e.g.*, grasping a wrong object, correcting a previous mistake). Moreover, workers pause some steps and resume them after others (*split*), skip necessary steps (*missing*), and swap the order from the prescribed one (*out-of-order*). These are kinds of order mistakes, yet we do not attach their labels explicitly because the video-text alignment reveals them. For example, if step 2 occurs before step 1, these swapped steps are considered *out-of-order*.

Mistake labels. If a segment indicates an execution mistake, we assign one label of the six execution mistake classes listed in Sec. 7.1. This categorization can be used to analyze mistake patterns across diverse tasks (*e.g.*, a commonly frequent category of mistakes). We observe multiple mistakes in just a few segments, so we label them as *Others* to avoid multi-label classification because the problem is difficult for current action understanding models.

Descriptions. We also provide the mistake segments with descriptions of why the performed actions are considered mistakes. They can be used to analyze model performance in detail or interact with workers through smart assistants. To maintain the quality of the descriptions, a template for each mistake class minimizes the use of modifiers, subjects, and articles. Table 8 shows description templates. The descriptions are written in English based on the templates, which are created based on mistake label classes. Figure 10 shows description examples annotated using the description templates.

7.6. Annotation Tool

Figure 11 shows an annotation web tool, which consists of four panes: video player, procedural text, annotation form, and annotation list. In the video player and procedural text panes, annotators can watch the video and read the procedural text. For video-text alignment annotations, we implemented a feature that allows them to move one second backward or forward in the video. In the annotation form pane, the annotators attach video-text alignment, mistake labels, and descriptions. For video-text alignments, annotators select the steps corresponding to the video segments. For mistake labels, annotators choose from predefined labels. For descriptions, annotators write explanations detailing why the segments are mistakes. In the annotation list pane, the annotators can view the annotation results. They can also edit or delete them by clicking on the buttons.

8. Dataset analysis

8.1. Inter-annotator agreement

We ensure the quality of the original annotations by testing for agreement with another annotator. Since annotating all of the samples is time-consuming, we randomly choose one out of 10 videos per task. The process involves the following two types of annotations by the new annotator. For video-text alignment, the annotator newly extracts segments and maps them to the corresponding steps. In addition, they watch the original segments and give new mistake labels and descriptions. To evaluate the annotation quality of video-text alignment, we calculate the temporal Intersection over the Union (tIoU) of segments. Note that we ignore the undefined steps for the calculation. We first calculate tIoU for each step, average by videos, and finally average them as an aggregated score. As for mistake labels and descriptions, we calculate Cohen's kappa [3] and BERTScore [40], respectively.

Results. We confirm that both annotations are high quality. For the video-text alignment, we achieve 88.8 tIoU, and for mistake labels and descriptions, we record 86.8 Cohen's kappa and 96.3 BERTScore. These high scores ensures the high agreement between the original and new annotations.

8.2. Statistics

8.2.1. Distribution of order mistakes

In Tab. 9, we provide the number of the four types of order mistakes listed in Sec. 7.1. The total count of order mistakes is 138. Looking at each type of error, *Pause-and-resume* is the most common type, occurring 51 times in total. The next most common one is *Out-of-order*, occurring 40 times in total. Looking at each task, the task where the mistakes happen most frequently is the cardboard crafts, with 61 errors. The top two types of mistakes in the task are *Pause-and-resume* and *Out-of-order*, which suggests that workers perform the steps in the instructions in a free order. In contrast, *Undefined-step* is the most common mistake in the color mixture experiments and the ionic reaction experiments, which suggests that extra steps are inserted. As such, each task shows different trends in terms of order mistakes.

8.2.2. Verb and noun distributions of mistake descriptions

Figure 12 shows the verb/nouns distributions of descriptions. We extract verbs and nouns using spaCy[5] based on `en_core_web_trf`. Note that for descriptions with the mistake labeled as *1. object* or *5. way*, we ignore the phrase after "but" because this phrase is not a mistake but rather a description of the correct step.

Regarding verbs, the top two verbs, "grasp" and "use," are the words in the templates. The third most frequent verb is "put," indicating mistakes related to the location and direction of objects (*e.g.*, "put batteries in the wrong direction" in the electrical circuits). This description can be confirmed in Fig. 10a. Regarding nouns, "liquid" and "plate" are the most frequent. These mistakes are common in color mixtures and ionic reactions (*e.g.*, "use yellow liquid but should use green liquid" in the color mixture in Fig. 10b. "Block" and "switch" are two nouns of the third most frequent, representing mistakes in building blocks and electrical circuits, respectively.

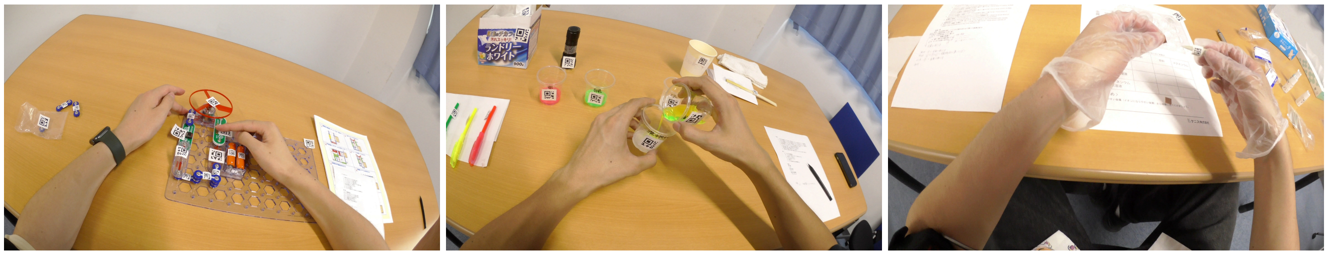
9. Dataset publication

We have published EgoOops to accelerate future studies for mistake action detection and reduce heavy burdens on

[5] https://spacy.io/

Table 8. Description templates.

Mistake label	Description
1. Object	Use {wrong object} but should use {correct object}
2. Mispick	Grasp {wrong object}
3. Correction	Correct error in step {step number}
4. Accident	{verb} (+{object})
5. Way	{verb} {object} {adverb} but should {verb} it/them {adverb}
6. Others	Free writing. Multiple sentences are allowed to describe multple mistakes.



(a) Described as "put batteries in the wrong direction" (mistake label: *Way*).

(b) Described as "use yellow liquid but should use green liquid" (mistake label: *Object*).

(c) Described as "drop copper plates on the table then pick them up with fingers" (mistake label: *Accident*).

Figure 10. Examples of mistake descriptions with video segments.

Table 9. The number of order mistakes.

Task	order mistakes				
	Missing	Out-of-order	Pause-and-resume	Undefined-step	Total
EC	2	10	12	6	30
CM	1	2	3	9	15
IR	2	4	1	6	13
BB	0	3	6	10	19
CB	7	21	29	4	61
All	12	40	51	35	138

workers due to mistakes. It can be obtained without a personal request in the expectation of effortless access.

9.1. Completeness of the relevant documentation

Section 7 describes how the data was collected and organized and what information it contains. We recommend users refer to the README document bundled with the published annotations, which explains how the dataset can be read.

9.2. Licensing and access

We regard accessibility and long-term preservation as important. We provide access to the dataset through the project page[6] hosted on the GitHub Pages[7]. We host the annotations on a GitHub repository[8]* because it presents an easy interface to update the data and show the history. We decided to host the videos on our laboratory's website[9]* due to a file size limitation of Github. We will provide these data with the necessary maintenance following our organization's policy [15].

EgoOops is licensed under CC BY-SA 4.0[10]. The license is a relatively permissive choice to accept wide uses because our intended use cases include commercial and industrial situations.

9.3. Consent and privacy

We asked the participants and obtained explicit consent to release the dataset that they had captured. Moreover, we managed to minimize the exposure of any personally identifiable information. Sounds are recorded but muted upon publication because workers sometimes spoke or talked to others. We trimmed unrelated segments to the task execution in the beginning and end of the videos (*e.g.*, other workers' help with task preparation and camera operation).

9.4. Ethics and responsible use

Our dataset is expected to be used for research on mistake detection. However, we do not prohibit the use of our dataset for other purposes as long as they are ethically acceptable. The entire part of our dataset was created for this study. Hence, it does not confront any ethical issues raised by other publicly released datasets.

[6]https://y-haneji.github.io/EgoOops-project-page/

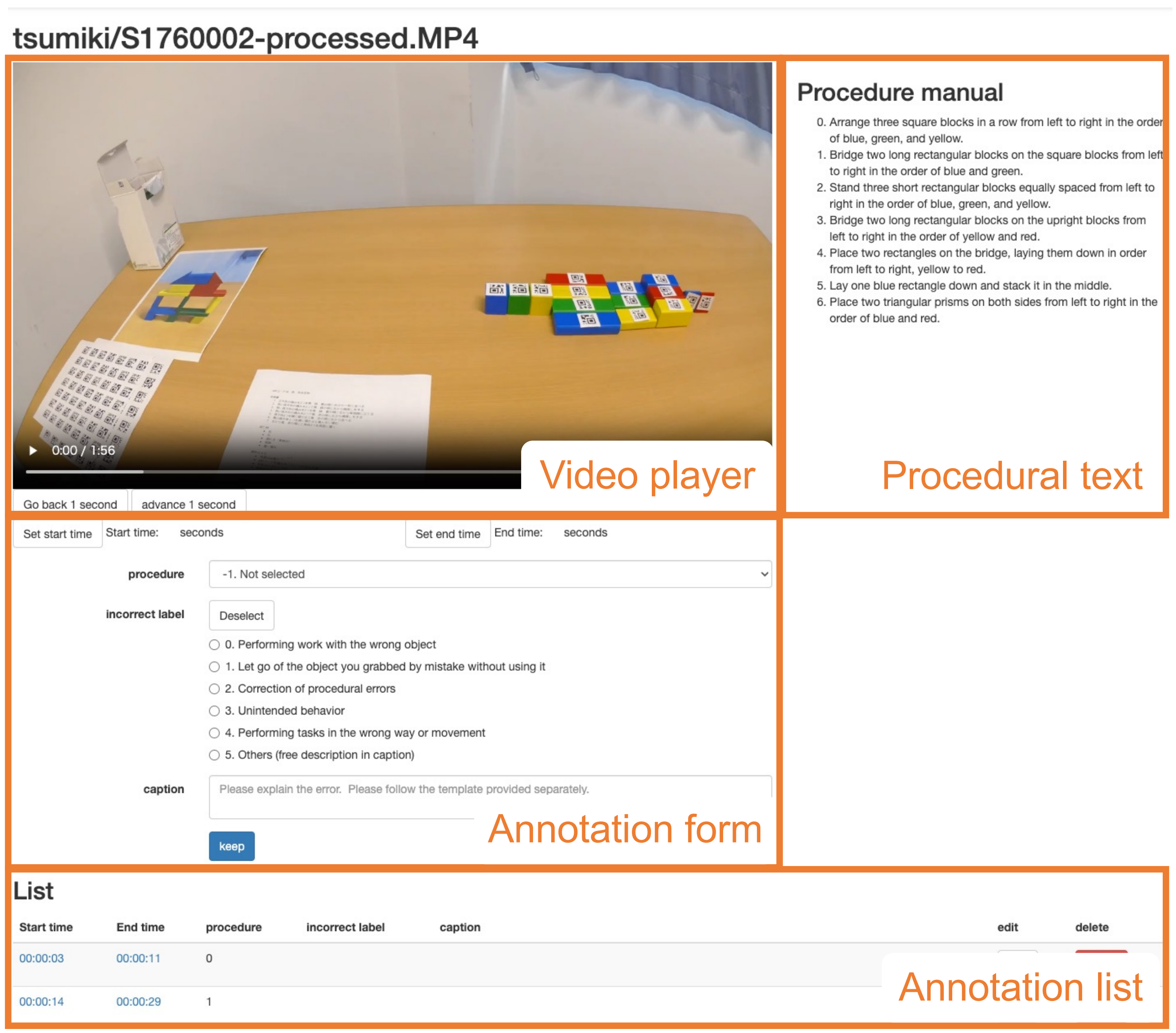
[7]https://pages.github.com

[8]https://github.com/Y-Haneji/EgoOops-annotations/

*Refer to the project page for the latest URLs.

[9]https://www.lsta.media.kyoto-u.ac.jp/resource/data/EgoOops/

[10]https://creativecommons.org/licenses/by-sa/4.0/


tsumiki/S1760002-processed.MP4

0:00 / 1:56

Go back 1 second | advance 1 second

Procedure manual

0. Arrange three square blocks in a row from left to right in the order of blue, green, and yellow.
1. Bridge two long rectangular blocks on the square blocks from left to right in the order of blue and green.
2. Stand three short rectangular blocks equally spaced from left to right in the order of blue, green, and yellow.
3. Bridge two long rectangular blocks on the upright blocks from left to right in the order of yellow and red.
4. Place two rectangles on the bridge, laying them down in order from left to right, yellow to red.
5. Lay one blue rectangle down and stack it in the middle.
6. Place two triangular prisms on both sides from left to right in the order of blue and red.

Set start time | Start time: seconds | Set end time | End time: seconds

procedure: -1. Not selected

incorrect label: Deselect

- 0. Performing work with the wrong object
- 1. Let go of the object you grabbed by mistake without using it
- 2. Correction of procedural errors
- 3. Unintended behavior
- 4. Performing tasks in the wrong way or movement
- 5. Others (free description in caption)

caption: Please explain the error. Please follow the template provided separately.

keep

List

Start time	End time	procedure	incorrect label	caption	edit	delete
00:00:03	00:00:11	0				
00:00:14	00:00:29	1				

Figure 11. Overview of annotation tool. An annotator can watch a video (upper left) and read a procedural text (upper right). In the annotation form (middle), the annotator attaches video-text alignment, mistake labels, and descriptions. The annotations are stored in the annotation list (bottom).

9.5. Legal compliance

We ensure our awareness and compliance with regional legal requirements.

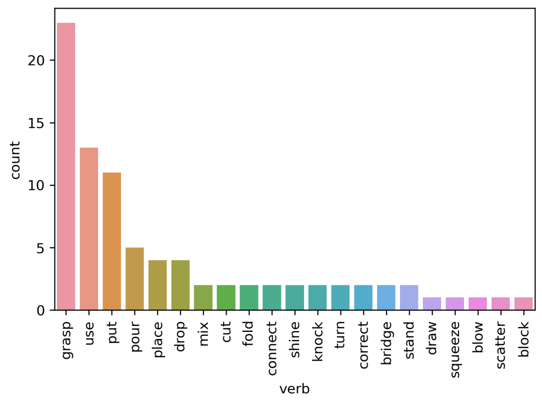
9.6. Author statement on responsibility

We state that we bear all responsibility in case of violation of rights, etc., and confirmation of the data license.

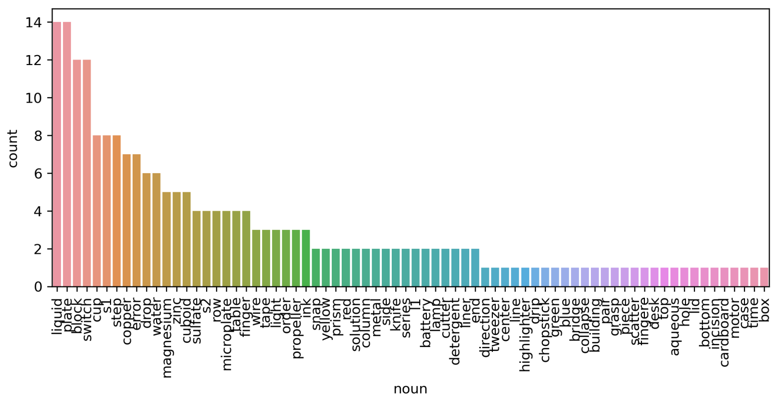
10. Details of experiments.

10.1. Mistake label classification

Existing mistake classifiers. We test existing mistake classifiers proposed by and trained on Assembly101 [31] and CaptainCook4D [24]. Sener *et al.* [31] proposed to apply a long-range video model TempAgg [30] using TSM [18] features. TSM is a video recognition model and they trained it for action recognition on Assembly101. The features are extracted using the completed TSM and passed to TempAgg for mistake classification into three classes: *correct*, *mistake*, and *correction*. We had not found their official imple-



(a) Distribution of verbs.



(b) Distribution of nouns.

Figure 12. Distribution of verbs and nouns in mistake descriptions.

mentation for feature extraction and mistake classification and thus, decided to reproduce them by ourselves.[11] As for a CaptainCook4D model, we chose the most performant variant of their error recognition models: a multi-layer perceptron head with a frozen 3D-ResNet [12] backbone for feature extraction. We followed their official implementations for feature extraction[12] and error recognition[13].

Multi-modal large language models (MLLMs). We evaluate two leading open-source MLLMs on EgoOops in a zero-shot manner aiming to inspect their visual reasoning capabilities; the two models are InternVL2.5-8B [2] and Qwen2-VL-7B-Instruct [37]. In our experiments, they solve mistake label classification given a trimmed video clip, the task's procedure, and the performed step. We construct the prompt by replacing placeholders in the template shown in Fig. 6. We designed this template with the following aim. 1. It contains the whole steps of the task to provide the context of the ongoing work. 2. It encourages them to discover mistake actions and their corrections actively. 3. It adopts multiple-choice questions as the direction format because MLLMs are trained to answer them well. The completed prompt and the frames sampled from the video clip are passed to the models as inputs, and the models output their answers in free-style texts. We used LMDeploy[14] for the inference processes to keep the experimental settings as fair as possible. For the pre-processing of the videos, we used their official implementations.[15][16] Specifically, for InternVL2.5, 24 frames are uniformly sampled from a video clip, and each of them is represented by 256 visual tokens; for Qwen2-VL, we sample each video at 2fps limiting the maximum number of frames to 48, and every frame is encoded to 128 visual tokens.

Evaluation metrics. We report standard metrics for mistake classification problems [8, 24, 29, 31, 38]: recall, precision, and F1 score. Since we aim to find mistake actions and their corrections, each metric is calculated only for the *mistake* and *correction* classes. Note that CaptainCook4D's model classifies each step into two classes: *normal* (*i.e.*, *correct*) and *error* (*i.e.*, *mistake*) [24], thus its evaluation results for the *correct* class are not available in our experiments. Also, we evaluate the predictions of MLLMs by checking whether the id and name of classes (*e.g.*, *0. correct*) exactly match the ground truths.

[11]We used the TSM model weights provided by Sener *et al.* at https://drive.google.com/file/d/11uFkqg1tWfWeATukjFL_HXHB6Gs1iGfU/view?usp=sharing.

[12]https://github.com/CaptainCook4D/feature_extractors/

[13]https://github.com/CaptainCook4D/error_recognition/

[14]https://lmdeploy.readthedocs.io/en/latest/index.html

[15]InternVL2.5: https://github.com/OpenGVLab/InternVL/

[16]Qwen2-VL: https://github.com/QwenLM/Qwen2-VL/