Open-ended Hierarchical Streaming Video Understanding with Vision Language Models

Supplementary Material

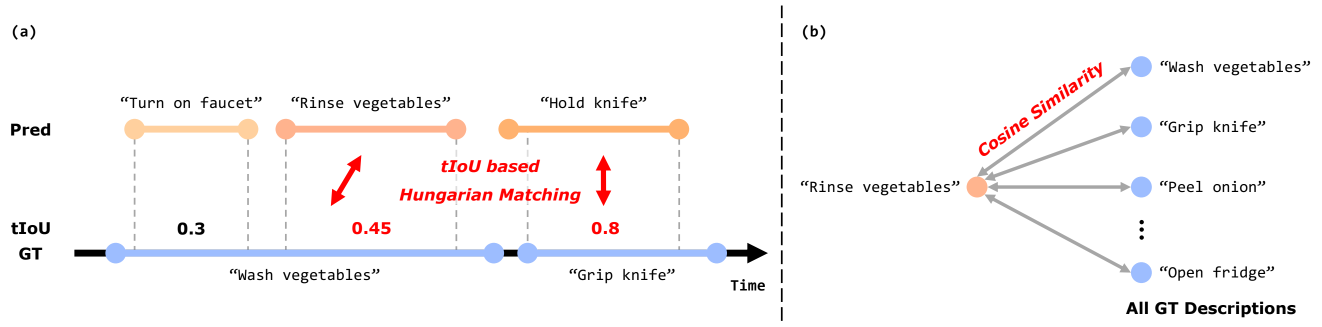

Figure 1. (a) $tIoU$-based Hungarian matching. Hungarian matching provides optimal one-to-one matching in terms of maximizing tIoU, unlike greedy matching which allows duplication. Here, the algorithm selects *"Rinse vegetables"* as the best match for the GT *"Wash vegetables"*. With a predefined tIoU threshold of 0.3, *"Rinse vegetables"* is considered a True Positive (TP), while the unmatched prediction *"Turn on faucet"* is classified as a False Positive (FP). If no prediction matches the GT or fails to meet the threshold, the instance is counted as a False Negative (FN). (b) The description *"Rinse vegetables"*, initially marked as a TP in F1 (loc.), is encoded into an embedding vector. Similarly, all GT descriptions in the test split are encoded, and pairwise cosine similarities are computed. If *"Wash vegetables"* appears among the top 5 most similar GT descriptions, *"Rinse vegetables"* is confirmed as a TP.

A. Details of Metrics

This section provides detailed explanations of the evaluation metrics presented in the main text.

Algorithm 1 get_hungarian_score in Python.

```
def get_hungarian_score(answer:list, prediction:list
    , iou_threshold=0.5):
    '''
    IN: answer[[st,ed],[st,ed]...],
    prediction[[st,ed],[st,ed]...]
    OUT: f1_score (float)
    '''
    profit = np.zeros((len(answer), len(prediction)))
    #calculate pairwise iou
    for i in range(len(answer)):
        for j in range(len(prediction)):
            profit[i][j] = calculate_iou(answer[i],
                prediction[j])
    #perform Hungarian Matching
    r, c = optimize.linear_sum_assignment(profit,
        maximize=True)
    tp = np.sum(np.where(profit[r, c] >=
        iou_threshold, 1, 0))
    a = answer.shape[0]
    p = prediction.shape[0]
    #return F1 score
    return 2*tp/(a+p)
```

Class-agnostic F1 [*F1 (loc.)*] Previous work [6] conducted an in-depth study on appropriate metrics for evaluating class-agnostic action proposals in a streaming setting, revealing that Hungarian F1 is an effective measure of performance. Accordingly, we use Hungarian F1 (hereafter, *F1 (loc.)*) to evaluate the streaming perception module. This metric employs the Hungarian matching algorithm [8], which provides optimal bipartite matching between class-agnostic ground truth $\{(t^s_m, t^e_m)\}_{m=1}^{M}$ and the model's predictions $\{(\hat{t}^s_m, \hat{t}^e_m)\}_{m=1}^{\hat{M}}$ in terms of $tIoU$. A prediction is considered a true positive if the overlap with the matched GT exceeds a predefined $tIoU$ (in our paper, $tIoU \in \{0.3, 0.5, 0.7\}$) threshold. Figure 1 (a) illustrates tIoU-based Hungarian matching and Algorithm 1 presents a brief Python implementation for calculating *F1 (loc.)*.

Method	Heir.	CIDEr	METEOR
GT Proposal	Step	17.7739	8.4363
	Substep	32.8570	10.7680
OpenHOUSE	Step	7.1885	4.1640
	Substep	14.3946	5.5151

Table 1. CIDEr, METEOR in EgoGS using evaluation tool [7].

Top-k F1 [*F1 (loc. + desc.)*] While *F1 (loc.)* only measures class-agnostic action proposals, our final output includes free-form descriptions of action instances. To account for this, we extend *F1 (loc.)* to define *F1 (loc. + desc.)*, adding a semantic relevance constraint for True Positives (TP) beyond the $tIoU$ condition. First, tIoU-based Hungarian matching is performed, identical to *F1 (loc.)*. Predictions that pass the $tIoU$ threshold are considered candidates. Each candidate prediction $\{s_p, e_p, d_p\}$ is matched to a ground truth (GT) instance $\{s_{gt}, e_{gt}, d_{gt}\}$ where s and e are the start and end times, and d represents the description. (Section 3.1)

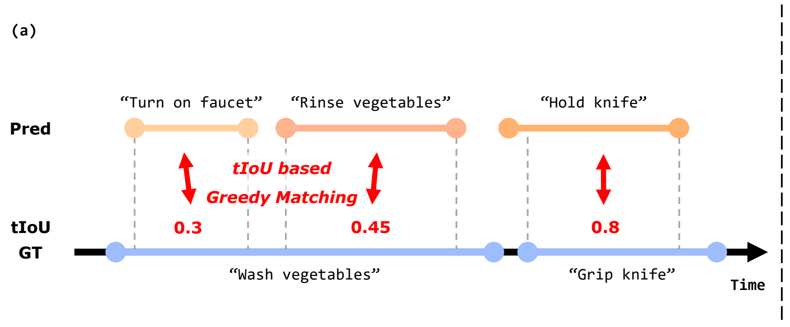

Figure 2. An example of calculating GPT, or BERTscore. With a $tIoU$ threshold of 0.3, the predictions *"Turn on faucet"* and *"Rinse vegetables"* are matched with the GT *"Wash vegetables"*, while *"Hold knife"* is matched with *"Grip knife"*. This duplicated matching with *"Wash vegetables"* highlights a key difference from the Hungarian F1 metric. After matching, GPT-Score or BERTscore is computed for each matched pair, and the final score is obtained by averaging all pair scores.

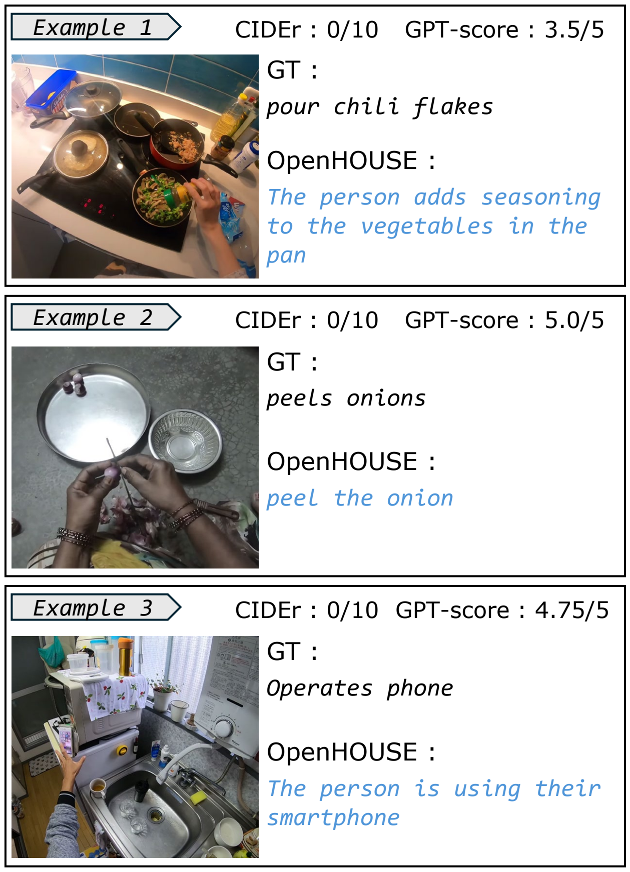
Example 1
CIDEr : 0/10 GPT-score : 3.5/5
GT :
pour chili flakes
OpenHOUSE :
The person adds seasoning to the vegetables in the pan
Example 2
CIDEr : 0/10 GPT-score : 5.0/5
GT :
peels onions
OpenHOUSE :
peel the onion
Example 3
CIDEr : 0/10 GPT-score : 4.75/5
GT :
Operates phone
OpenHOUSE :
The person is using their smartphone

Figure 3. Examples of CIDEr [14] and GPT-Score [9] results from EgoGS dataset. Each score was computed at tIoU 0.3. GPT-Score reflects the average of four semantic dimensions: CU, CO, DO, and TU.

The final step is to determine whether d_p and d_{gt} match or not. Since d_p is a free-form output from a powerful large-scale VLM, it tends to be highly detailed, making semantic relevance crucial—something traditional N-gram-based metrics fail to capture. To effectively evaluate the semantic relevance of the matched prediction, we use a zero-shot evaluation approach inspired by CLIP [11]. For each matched prediction, we encode the candidate description d_p and all GT descriptions $\{d_m\}_{m=1}^{M}$ using a GPT-4 [1] text encoder, where M represents the total number of GT descriptions in the whole test split. Pairwise cosine similarity is then calculated between d_p and all GT descriptions, which are subsequently ranked by similarity. If the matched d_{gt} appears in the top-k rankings, the prediction is finally classified as a TP. Figure 1 (b) provides an example of this process. For all experiment, we choose $k = 5$ as we empirically found that at least five ground truth descriptions often share near-identical semantics.

Description Quality Metrics In addition to the main metrics, we use another metrics to evaluate the generated descriptions: GPT-Score [9] and BERTscore [15], as introduced in Section 4 of the main text. For GPTScore, we follow [9], scoring all true positive predictions on a 1–5 scale across the CI (Correctness of Information), DO (Detailed Orientation), CU (Contextual Understanding), and TU (Temporal Understanding) categories using GPT-3.5. The CO (Consistency) category is excluded as it is not suitable for evaluating hierarchical action descriptions. Detailed prompts used for evaluating the CI, DO, CU, and TU categories can be found in Prompt 1, 2, 3, 4. Unlike *F1 (loc.)*, which identifies optimal one-to-one matching between ground truth and predictions, these metrics use *greedy matching*, allowing multiple predictions to match the same ground truth. A predefined $tIoU$ threshold is applied to filter predictions before calculating the scores.

N-gram based metric We found that popular N-gram-based methods (e.g. CIDEr [14]) are not suitable for evaluating the caption quality of OpenHOUSE. Since we use a powerful VLM in a zero-shot setting, it often produces detailed descriptions that may align even better with the actual action than the ground truth, but may not share the exact

Prompt 1 `CI Prompt` for Evaluation.

```
role: system,
content:
   You are an intelligent chatbot designed for evaluating
        the factual accuracy of generative outputs for video
        -based question-answer pairs.
   Your task is to compare the predicted answer with the
        correct answer and determine if they are factually
        consistent. Here's how you can accomplish the task:
   ------
   ##INSTRUCTIONS:
   - Focus on the factual consistency between the predicted
        answer and the correct answer. The predicted answer
        should not contain any misinterpretations or
        misinformation.
   - The predicted answer must be factually accurate and
        align with the video content.
   - Consider synonyms or paraphrases as valid matches.
   - Evaluate the factual accuracy of the prediction
        compared to the answer.

role: user,
content:
   Please evaluate the following video-based question-answer
         pair:"
   Question: {question}
   Correct Answer: {answer}
   Predicted Answer: {pred}
   Provide your evaluation only as a factual accuracy score
        where the factual accuracy score is an integer value
         between 0 and 5, with 5 indicating the highest
        level of factual consistency.
   Please generate the response in the form of a Python
        dictionary string with keys 'score', where its value
         is the factual accuracy score in INTEGER, not
        STRING.
   DO NOT PROVIDE ANY OTHER OUTPUT TEXT OR EXPLANATION. Only
         provide the Python dictionary string.
   For example, your response should look like this: {''
        score': 4.8}.
```

Prompt 2 `DO Prompt` for Evaluation.

```
role: system,
content:
   You are an intelligent chatbot designed for evaluating
        the detail orientation of generative outputs for
        video-based question-answer pairs.
   Your task is to compare the predicted answer with the
        correct answer and determine its level of detail,
        considering both completeness and specificity. Here'
        s how you can accomplish the task:
   ------
   ##INSTRUCTIONS:
   - Check if the predicted answer covers all major points
        from the video. The response should not leave out
        any key aspects.
   - Evaluate whether the predicted answer includes specific
         details rather than just generic points. It should
        provide comprehensive information that is tied to
        specific elements of the video.
   - Consider synonyms or paraphrases as valid matches.
   - Provide a single evaluation score that reflects the
        level of detail orientation of the prediction,
        considering both completeness and specificity.

role: user,
content:
   Please evaluate the following video-based question-answer
         pair:
   Question: {question}
   Correct Answer: {answer}
   Predicted Answer: {pred}
   Provide your evaluation only as a detail orientation
        score where the detail orientation score is an
        integer value between 0 and 5, with 5 indicating the
         highest level of detail orientation.
   Please generate the response in the form of a Python
        dictionary string with keys 'score', where its value
         is the detail orientation score in INTEGER, not
        STRING.
   DO NOT PROVIDE ANY OTHER OUTPUT TEXT OR EXPLANATION. Only
         provide the Python dictionary string.
   For example, your response should look like this: {''
        score': 4.8}.
```

Prompt 3 `CU Prompt` for Evaluation.

```
role: system,
content:
   You are an intelligent chatbot designed for evaluating
        the contextual understanding of generative outputs
        for video-based question-answer pairs.
   Your task is to compare the predicted answer with the
        correct answer and determine if the generated
        response aligns with the overall context of the
        video content. Here's how you can accomplish the
        task:
   ------
   ##INSTRUCTIONS:
   - Evaluate whether the predicted answer aligns with the
        overall context of the video content. It should not
        provide information that is out of context or
        misaligned.
   - The predicted answer must capture the main themes and
        sentiments of the video.
   - Consider synonyms or paraphrases as valid matches.
   - Provide your evaluation of the contextual understanding
         of the prediction compared to the answer.

role: user,
content:
   Please evaluate the following video-based question-answer
         pair:
   Question: {question}
   Correct Answer: {answer}
   Predicted Answer: {pred}
   Provide your evaluation only as a contextual
        understanding score where the contextual
        understanding score is an integer value between 0
        and 5, with 5 indicating the highest level of
        contextual understanding.
   Please generate the response in the form of a Python
        dictionary string with keys 'score', where its value
         is contextual understanding score in INTEGER, not
        STRING.
   DO NOT PROVIDE ANY OTHER OUTPUT TEXT OR EXPLANATION. Only
         provide the Python dictionary string.
   For example, your response should look like this: {''
        score': 4.8}.
```

Prompt 4 `TU Prompt` for Evaluation.

```
role: system,
content:
   You are an intelligent chatbot designed for evaluating
        the temporal understanding of generative outputs for
         video-based question-answer pairs.
   Your task is to compare the predicted answer with the
        correct answer and determine if they correctly
        reflect the temporal sequence of events in the video
         content. Here's how you can accomplish the task:
   ------
   ##INSTRUCTIONS:
   - Focus on the temporal consistency between the predicted
         answer and the correct answer. The predicted answer
         should correctly reflect the sequence of events or
        details as they are presented in the video content.
   - Consider synonyms or paraphrases as valid matches, but
        only if the temporal order is maintained.
   - Evaluate the temporal accuracy of the prediction
        compared to the answer.

role: user,
content:
   Please evaluate the following video-based question-answer
         pair:
   Question: {question}
   Correct Answer: {answer}
   Predicted Answer: {pred}
   Provide your evaluation only as a temporal accuracy score
         where the temporal accuracy score is an integer
        value between 0 and 5, with 5 indicating the highest
         level of temporal consistency.
   Please generate the response in the form of a Python
        dictionary string with keys 'score', where its value
         is the temporal accuracy score in INTEGER, not
        STRING.
   DO NOT PROVIDE ANY OTHER OUTPUT TEXT OR EXPLANATION. Only
         provide the Python dictionary string.
   For example, your response should look like this: {''
        score': 4.8}.
```

vocabulary with the ground truth. This is not a flaw to penalize but rather aligns with our ultimate goal of open-ended understanding. However, N-gram-based methods require exact word matches, which unjustly penalizes our more detailed descriptions.

Figure 3 illustrates some examples of this. Unlike N-gram based method (CIDEr [14]), we found that the metrics that leverage large-scale language models, such as GPT-based scoring (GPT-Score [9]), offers a more robust evaluation framework for semantic similarity. Note that our main metric, *F1 (loc. + desc.)*, also uses a powerful VLM's text encoder [1] for encoding the descriptions into vector representations, effectively capturing semantic correspondence. For completeness, we provide results of N-gram based metric in Table 1.

B. Utilization of Different VLMs

One of the key components of our system is the VLM, which generates free-form descriptions. Table 3 shows the variation in caption quality with different VLMs. The results indicate that while more advanced VLMs generally improve performance, there is a point of diminishing gains once the VLM is sufficiently powerful. This highlights the inherent complexity of Hierarchical Streaming Video Understanding; further improvements cannot rely solely on VLM advancements—enhancing the streaming module is also crucial for meaningful progress.

C. Generating descriptions for incomplete action instances

In Section 4 of the main text, we reported results where the streaming module invoked the VLM to predict substeps and steps at the points when each action instance ended, and to predict the overall goal when the video finished. However, in real world scenarios, it is crucial to generate predictions even when an action instance or video is still in progress.

Thus, in this section, we present the prediction results for substep instances, step instances, and the entire video at various completion stages: 25%, 50%, 75%, and 100%. This analysis aims to demonstrate the OpenHOUSE's ability to provide reliable predictions for substeps, steps, and goals even before an action instance or the entire video reaches completion.

Table 2 presents the VLM inference results on both ground truth (GT) temporal annotations and those generated by the OpenHOUSE streaming module at different completion stages. While there is a clear trend of improved performance with increased observation, the results at 50% completion show only slight degradation compared to full observation. Note that we only have 134 test samples for goal accuracy, so a single correct or incorrect prediction can lead to approximately a 1% fluctuation in performance, explaining the observed perturbations.

These findings highlight that OpenHOUSE can provide reliable inferences even when only partial information is available, emphasizing the potential of OpenHOUSE in real-world, online scenarios where actions are often incomplete.

Method	**Completion**	**Hier.**	**F1 (loc. + desc.) ↑**			**Goal Acc. ↑**
			0.3	0.5	0.7	
GT	25%	Step	18.93	18.93	18.93	39.55
		Substep	23.66	23.66	23.66	
	50%	Step	24.56	24.56	24.56	45.52
		Substep	28.52	28.52	28.52	
	75%	Step	26.53	26.53	26.53	47.76
		Substep	31.78	31.78	31.78	
	100%	Step	28.68	28.68	28.68	47.01
		Substep	33.12	33.12	33.12	
OpenHouse	25%	Step	9.68	7.86	5.36	44.03
		Substep	13.55	11.14	8.05	
	50%	Step	12.84	10.66	7.45	42.54
		Substep	16.99	13.93	9.91	
	75%	Step	14.36	12.02	8.38	45.52
		Substep	18.8	15.41	10.66	
	100%	Step	15.23	12.67	8.680	47.76
		Substep	19.79	16.11	10.89	

Table 2. Experiments on EgoGS validating description generation for incomplete action instances.

D. Dataset Analysis

In Section D, we aim to discuss the comparison between the Ego4D GoalStep (EgoGS) dataset and the Ego4D GoalStep pseudo (EgoGS pseudo) dataset generated through our dataset generation pipeline. Additionally, we provide statistics on the reconstructed hierarchical datasets, including Ego-Exo4D Keystep (EgEx) and Epic-Kitchens 100 (EK100).

Comparison between EgoGS and EgoGS pseudo Figure 4 (a) illustrates a comparison of step segment durations between EgoGS and EgoGS pseudo. The step segment duration for EgoGS averages 50.69 seconds, slightly differing from the original value reported in the Ego4D GoalStep dataset [12] because only the train and validation data, excluding the test dataset, were used for this calculation. EgoGS pseudo has an average step segment duration of 46.08 seconds, which is similar to EgoGS. Additionally, the overall distribution of step segment durations for EgoGS pseudo aligns well with that of EgoGS. This demonstrates that the dataset generated through our pipeline effectively approximates the original dataset.

Dataset Statistics Figure 4 (b), (c), (d) respectively illustrate the distributions of goal, step, and substep segment durations for EgoGS pseudo, EgEx, and EK100. The goal, step, and substep segments of EgoGS pseudo have average durations of 1532.32 seconds, 46.08 seconds, and 22.44 seconds, respectively. For EgEx, the goal, step, and substep segments have average durations of 308.09 seconds, 32.69

Model	Hier.	F1 (loc. + cap.) ↑			GPT-Score ↑ [9]				BERTScore ↑ [15]
		0.3	0.5	0.7	CI	DO	CU	TU	
GPT-4o [10]	Step	15.53	12.65	8.59	3.33	2.688	3.559	2.852	0.857
	Substep	19.51	16.15	11.19	3.027	2.595	3.336	2.706	0.866
InternVL2-40B-AWQ [3]	Step	15.23	12.67	8.68	3.192	2.606	3.381	2.670	0.858
	Substep	19.79	16.11	10.89	2.869	2.559	3.164	2.543	0.873
InternVL2-8B [3]	Step	10.01	8.21	5.49	2.791	2.374	3.053	2.222	0.843
	Substep	13.09	10.65	7.57	2.245	2.248	2.644	1.978	0.862

Table 3. Experimental results on the EgoGS dataset using different VLMs. Here, CI, DO, CU, TU in GPT-Score refer to "Correctness of Information", "Detail Orientation", "Contextual Understanding", "Temporal Understanding" respectively.

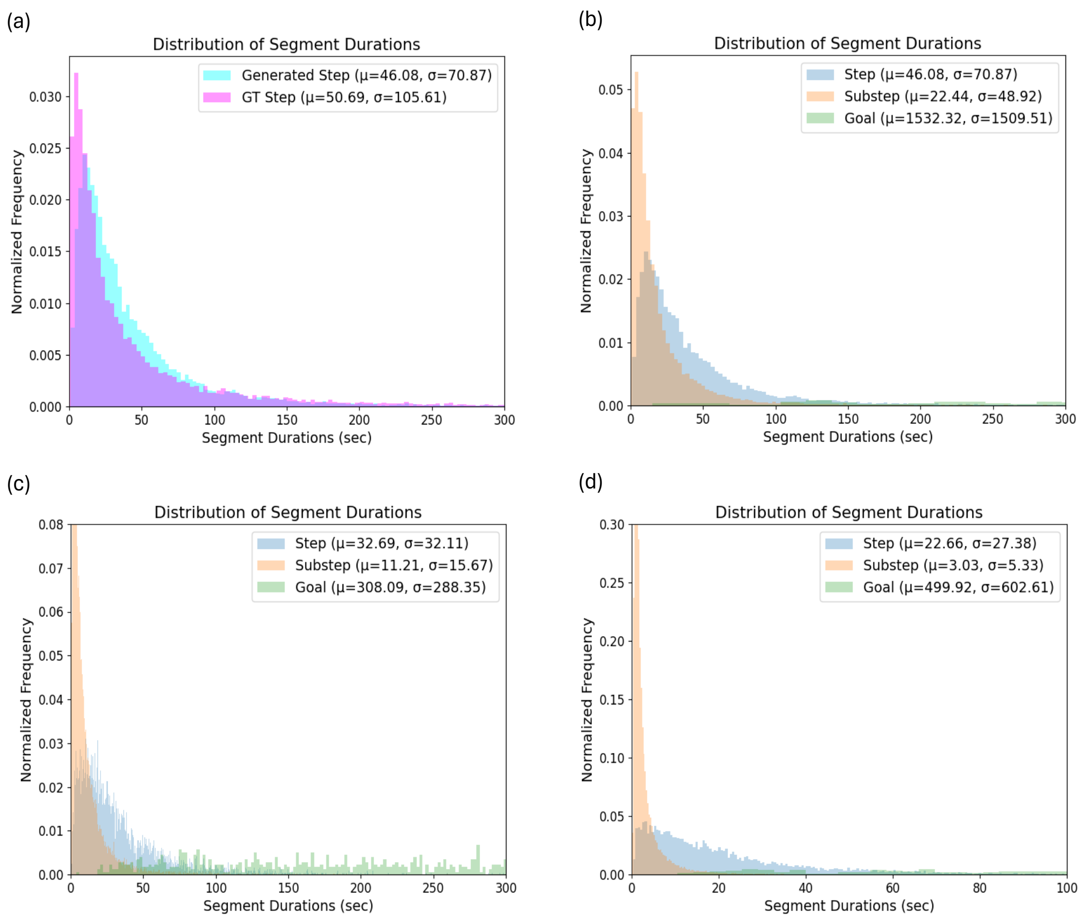

Figure 4. Dataset statistics

seconds, and 11.21 seconds, respectively. The goal, step, and substep segments of EK100 have average durations of 499.92 seconds, 22.66 seconds, and 3.03 seconds, respectively.

Human Validators in Dataset Generation Five human annotators participated in the dataset annotation and validation process. Each annotator contributed approximately 20 hours of work and was compensated $200.

E. VLM inference details

In section E, we describe the details of the inference process for each level in the hierarchy: *substep*, *step*, and *goal*, using VLM. Additionally, we provide comprehensive details regarding inference speed measurement

Substep Inference Each *substep* is inferred using the following inputs: (i) video frames sampled at 1-second intervals from the corresponding *substep* instance, and (ii) text predictions from prior *substeps* within the same step. With these inputs, predictions are made using the prompt 7. The generated short form response serves as the prediction re-

Prompt 5 `Goal Prompt` for VLM inference.

```
I am planning to add annotations to a video. The annotations
      form a three-level hierarchy: goal, steps, and
     substeps. Here are the specific requirements:
1. Goal Annotation: There is only one goal annotation for
     the entire video.
2. Step Annotations: Each step annotation is ordered
     chronologically and must follow the completion of all
     substeps within the previous step. For example, Step 2
     cannot begin until all substeps of Step 1 are completed
     .
3. Substep Annotations: These are specific parts of the
     video that detail the actions within each step.

Given an image sequence extracted from a video, predict the
     most appropriate goal for the video based on the frames
      from each step and the short form responses for each
     step.
The short form responses of the steps are provided as a time
     -ordered list in text format, where the 0th index is
     the earliest step and higher indices represent more
     recent steps.
Utilize this context to predict the overall goal of the
     video.

Generate a response:
The response should consist of a single sentence that
     succinctly describes the goal.
Use the following list of short form responses for each step
      (in text format and time-ordered):
Short form response of step: {short_form_step}

Output format must be:
Answer: (goal)
```

Prompt 6 `Step Prompt` for VLM inference.

```
I am planning to add annotations to a video. The annotations
      form a three-level hierarchy: goal, steps, and
     substeps. Here are the specific requirements:
1. Goal Annotation: There is only one goal annotation for
     the entire video.
2. Step Annotations: Each step annotation is ordered
     chronologically and must follow the completion of all
     substeps within the previous step. For example, Step 2
     cannot begin until all substeps of Step 1 are completed
     .
3. Substep Annotations: These are specific parts of the
     video that detail the actions within each step.

Given an image sequence extracted from a video clip, predict
      the most appropriate step occurring in the clip based
     on the sequence and the previous steps.
The previous steps are provided as a time-ordered list of
     long form responses in text format, where the 0th index
      is the earliest step and higher indices represent more
      recent steps.
If there are no previous steps, the list will be empty. If
     there are more than 10 previous steps, only the 10 most
      recent responses will be provided. Utilize this
     context to improve the prediction for the current step.

First, Generate two types of responses:
Short form response: A single sentence that succinctly
     describes the step.
Long form response: A detailed and accurate description of
     the step based on the image sequence, considering the
     previous steps if provided.

After generating the responses, revise the long form
     response to ensure it aligns with the short form
     response for consistency.

Use the following list of previous long form responses in
     text format to ensure continuity and logical
     progression (the list may be empty if there are no
     prior steps, and a maximum of the 10 most recent
     responses will be provided):
Previous long form response: {prediction_list}

Output format must be:
Answer:
short form response: (response)
long form response (before revision): (response)
long form response (after revision): (response)
```

Prompt 7 `Substep Prompt` for VLM inference.

```
I am planning to add annotations to a video. The annotations
      form a three-level hierarchy: goal, steps, and
     substeps. Here are the specific requirements:
1. Goal Annotation: There is only one goal annotation for
     the entire video.
2. Step Annotations: Each step annotation is ordered
     chronologically and must follow the completion of all
     substeps within the previous step. For example, Step 2
     cannot begin until all substeps of Step 1 are completed
     .
3. Substep Annotations: These are specific parts of the
     video that detail the actions within each step.

Given an image sequence extracted from a video clip, predict
      the most appropriate substep occurring in the clip
     based on the sequence and the previous substeps of the
     current step.
The previous substeps are provided as a time-ordered list of
      long form responses in text format, where the 0th
     index is the earliest substep and higher indices
     represent more recent substeps.
If there are no previous substeps, the list will be empty.
     Utilize this context to improve the prediction for the
     current substep.

First, Generate two types of responses:
Short form response: A single sentence that succinctly
     describes the substep.
Long form response: A detailed and accurate description of
     the substep based on the image sequence, considering
     the previous substeps if provided.

After generating the responses, revise the long form
     response to ensure it aligns with the short form
     response for consistency.

Use the following list of previous long form responses in
     text format to ensure continuity and logical
     progression (the list may be empty if there are no
     prior substeps):
Previous long form response: {prediction_list}

Output format must be:
Answer:
short form response: (response)
long form response (before revision): (response)
long form response (after revision): (response)
```

sult, while the long form response (after revision) is used as input for predicting the next *substep*.

Step Inference For step-level inference, the input consists of: (i) images sampled from the *substep* instances within the given *step* at 3.3-second intervals, and (ii) text predictions from up to 10 previous *steps*. With these inputs, predictions are made using the prompt 6. Again, the short form response represents the prediction result, while the long form response (after revision) is employed to predict the subsequent *step*.

Goal Inference *Goal* inference utilizes: (i) a single image per *step*, and (ii) text predictions from all the *steps*. With these inputs, predictions are made using the prompt 5.

Inference Speed Measurement Details As discussed in Section 4.2, we evaluated the inference speed of a video with 2758 * 16 frames (46 minutes at 16 fps) using the Intern VL2-40B-AWQ [3] model. The measurement was conducted utilizing 4 * RTX 3090 GPUs. The measured FPS represents the model's average processing FPS.

Following the technical details in [2], the reported 24 FPS includes the *entire OpenHOUSE pipeline*, encompassing: (i) online feature extraction from raw frames, (ii) Streaming Module inference, and (iii) VLM inference.

F. Implementation Details

As discussed in Section 3.3.1 of the main paper, our Streaming Module consists of three heads: a state-emitting head, a progression head for steps, and a progression head for substeps. Each head shares an RNN backbone comprising three recurrent layers with a hidden state size of 768.

The state-emitting head is trained using the standard cross-entropy loss, following the approach in [6]. The progression heads for both steps and substeps are trained using the histogram loss described in [4], configured with 10 bins and a standard deviation (σ) of 0.15.

We trained using AdamW optimizer with a learning rate of 3e-4, a batch size of 16, and a weight decay of 0.01. The model is trained for a total of 30 epochs.

G. Streaming Module Comparison

Since our streaming module is based on the ActionSwitch [6] design, we conducted apple-to-apple evaluations on EK-100 against various class-agnostic On-TAL baselines. (Table 4). Key observations are: (i) Streaming Module (SM) without hybrid detection performs comparably to ActionSwitch, validating our design choice, (ii) SM with hybrid strategy significantly outperforms prior methods, setting a new SOTA in class-agnostic On-TAL. These results further confirm the effectiveness and superiority of our hybrid action boundary detection method.

Method	F1@0.5	Precision@0.5	Recall@0.5
CAG-QIL [5]	23.117	21.347	25.206
SimOn [13]	4.395	2.351	33.481
OAD-Grouping [5]	21.416	25.533	18.442
ActionSwitch [6]	32.444	29.858	35.519
SM w/o Hybrid	31.459	38.689	26.506
SM (OpenHOUSE)	**48.954**	**48.172**	**49.763**

Table 4. SM(Streaming Module) comparison in EK100 (Class-agnostic metrics)

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. 2, 4

[2] Joungbin An, Hyolim Kang, Su Ho Han, Ming-Hsuan Yang, and Seon Joo Kim. Miniroad: Minimal rnn framework for online action detection. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023. 6

[3] Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821*, 2024. 5, 6

[4] Jesse Farebrother, Jordi Orbay, Quan Vuong, Adrien Ali Taïga, Yevgen Chebotar, Ted Xiao, Alex Irpan, Sergey Levine, Pablo Samuel Castro, Aleksandra Faust, et al. Stop regressing: Training value functions via classification for scalable deep rl. *arXiv preprint arXiv:2403.03950*, 2024. 7

[5] Hyolim Kang, Kyungmin Kim, Yumin Ko, and Seon Joo Kim. Cag-qil: Context-aware actionness grouping via q imitation learning for online temporal action localization. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 7

[6] Hyolim Kang, Jungsuk Hyun, Joungbin An, Youngjae Yu, and Seon Joo Kim. Actionswitch: Class-agnostic detection of simultaneous actions in streaming videos. In *European Conference on Computer Vision (ECCV)*, 2024. 1, 7

[7] Ranjay Krishna, Kenji Hata, Frederic Ren, Li Fei-Fei, and Juan Carlos Niebles. Dense-captioning events in videos. In *Proceedings of the IEEE international conference on computer vision*, pages 706–715, 2017. 1

[8] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955. 1

[9] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-chatgpt: Towards detailed video understanding via large vision and language models. *arXiv preprint arXiv:2306.05424*, 2023. 2, 4, 5

[10] OpenAI. Hello gpt-4o, 2024. 5

[11] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021. 2

[12] Yale Song, Eugene Byrne, Tushar Nagarajan, Huiyu Wang, Miguel Martin, and Lorenzo Torresani. Ego4d goal-step: Toward hierarchical understanding of procedural activities. *Advances in Neural Information Processing Systems*, 2024. 4

[13] Tuan N Tang, Jungin Park, Kwonyoung Kim, and Kwanghoon Sohn. Simon: A simple framework for online temporal action localization. *arXiv preprint arXiv:2211.04905*, 2022. 7

[14] Ramakrishna Vedantam, C Lawrence Zitnick, and Devi Parikh. Cider: Consensus-based image description evaluation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 2, 4

[15] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019. 2, 5