Token-Efficient Video Game Summarization for Multimodal LLMs

Christian Calvo **Sunggyeol Oh** **Ermias Asnake** **Junwoo Seo** **Siddharth Rakshit**
Virginia Tech
{kuzoto, sunggyeol, easnake, junwoo, sidrak5}@vt.edu
https://github.com/Kuzoto/Apex_AI

1 Problem statement

Video analysis and summarization through a Large-Language Model (LLM) is traditionally a resource-intensive process. When an LLM is asked to interpret full video frame data, it must process a large number of tokens. This significantly increases the computational workload of the model, raising power consumption, slowing inference, and driving up operating costs. As a result, large-scale or continuous video analysis becomes impractical and unnecessarily expensive.

A major contributor to this inefficiency is that LLMs typically analyze every frame equally, regardless of whether the frame contains relevant activity or no meaningful information at all. This creates substantial token waste. However, if the system can determine which specific events occur in each frame and use that event information as guiding context, the LLM can focus its attention on only the frames that matter. Under this approach, we anticipate that summary quality will remain consistent, while token use, model workload, and overall cost decrease dramatically. This idea suggests a path toward more efficient, context-driven video understanding systems. Recent advances in vision-language models have made video understanding more accessible, but the fundamental challenge of token efficiency remains unaddressed. While some approaches attempt to learn frame selection policies through training, they require extensive datasets and computational resources. Our hypothesis is that leveraging external structured data, readily available in many domains, offers a more practical and immediately deployable solution.

Video games provide a particularly useful domain for testing this hypothesis because they generate precise, structured event data. Through Overwolf's API, developers can access real-time game events with accurate timestamps, allowing them to detect exactly when significant actions occur. This makes it possible to pair each meaningful moment in gameplay with a screenshot or video segment and an accompanying list of events.

The goal of our project is to evaluate this concept using gameplay recordings and screenshots. By using Overwolf to capture images at the exact moments when events occur, and by generating a detailed event list tied to each image, we can supply an LLM with both the visual input and the contextual event information. We then analyze how well the LLM summarizes the visual content when supported by this targeted context, and how the required tokens compare to traditional frame-by-frame analysis.

Ultimately, our objective is to demonstrate that incorporating event-driven context into LLM-based video summarization is a promising direction for future research. If successful, this approach could lead to more efficient, scalable, and cost-effective models that intelligently prioritize keyframes, reducing redundancy without sacrificing accuracy.

2 What you proposed vs. what you accomplished

- ~~Acquire and pre-process data~~: This step mostly consisted of collecting the sample data that we used to generate summaries. This is because without needing to train the object recognition model, we really only needed sample data. So much of this effort transitioned to step 3.

- *Train object recognition model*: This step was never completed since it was abandoned based on the feedback received in our proposal.

- ~~Develop screenshot and event-logging program~~ This was expanded upon to also include recording a video alongside the event screenshot and log to be used to generate a baseline summary.

- ~~Integrate LLM and TTS functionality~~

- ~~Analyze results and efficiency~~: This was accomplished differently than what we proposed. Rather than basing our analysis on human judgement, we used an LLM-as-a-Judge. (https://claude.ai/share/ccfa4423-b5b1-48e9-a572-6ee8b0f7c09d)

3 Related work

The rapid advancement of vision-language models has established a foundational paradigm for multi-modal understanding. (Dosovitskiy et al., 2021) introduced Vision Transformers (ViT), demonstrating that pure transformer-based architectures effectively learn visual representations by treating images as sequences of patches. Building on this, (Radford et al., 2021) introduced CLIP, establishing that language provides powerful supervision for visual understanding at scale. These foundational works enabled the development of instruction-tuned systems like (Li et al., 2022) (BLIP), which unifies vision-language understanding and generation, and (Liu et al., 2023) (LLaVA), which transfers instruction-following capabilities to vision-language models.

Recent work has extended these approaches to video understanding. (Alayrac et al., 2022) introduced Flamingo, a few-shot learner that processes interleaved visual inputs and text, establishing key mechanisms for handling multiple visual frames in sequence. Following this, (Maaz et al., 2023) developed Video-ChatGPT for detailed video understanding through open-ended conversation, and (Chen et al., 2023) proposed VideoLLM leveraging LLM sequence reasoning for video analysis. (Dai et al., 2023) advanced instruction-following in vision-language models, while (Liu et al., 2021) proposed Swin Transformers for hierarchical visual understanding.

A critical challenge emerges when scaling to video: converting long frame sequences into tokens incurs significant computational costs and exceeds context windows. (Gao et al., 2023) addressed this for video question answering through selective spatial-temporal aggregation. To mitigate token explosion, (Chen et al., 2024) introduced VideoLLM-online with its LIVE framework that learns when to process frames and when to remain silent, effectively filtering redundant content. (Weng et al., 2024) proposed LongVLM for efficient long video understanding through improved encoding. Most relevant, (Qian et al., 2024) presented VideoStreaming with Adaptive Memory Selection, which selects question-relevant frames to reduce redundancy in a context-aware manner.

Our work differs by leveraging external event data as an explicit, efficient frame selection proxy. While prior methods learn adaptive selection based on queries or streaming dialogue contexts, we hypothesize that structured domains like video games benefit significantly from event-driven selection, capturing frames precisely when meaningful game events occur. This out-of-band approach, using APIs like Overwolf to capture event-triggered frames, pre-filters video content before LLM processing, potentially reducing tokens without sacrificing analysis quality or detail. We validate this approach for game-specific video summarization, where structured event data is naturally available and highly informative.

4 Dataset

We construct a self-collected, multi-modal dataset of Apex Legends gameplay using the Overwolf API.[1] The corpus comprises 33 *collections*, each corresponding to approximately 10 seconds of continuous gameplay recorded on November 19, 2025. Every collection contains three synchronized modalities:

- **Images.** In-game screenshots captured in real time via Overwolf, triggered by specific gameplay events such as damage, kills, knockdowns, or item interactions.

- **Text.** A structured event log emitted by Overwolf over the same time window. The log consists of an ordered sequence of game events (e.g., weapon swaps, inventory updates, player state changes). Events are ordered but not explicitly timestamped.

- **Video.** A continuous gameplay recording covering the full match. Each collection is associated with a corresponding video segment,

[1] https://github.com/Kuzoto/Apex_AI/tree/main/sample_data

which we treat as a qualitative reference and a potential baseline for future video-centric models.

Our task is to generate natural language summaries that describe what happens within each collection, combining surface-level actions (e.g., dealing damage with a particular weapon, looting, or healing) with higher-level tactical intent (e.g., repositioning, preparing for an engagement, third-partying another squad). This setting is challenging for several reasons: (i) the event log is sparse and event-driven, recording only state changes rather than continuous actions; (ii) the log format is noisy, with nested and sometimes double-encoded JSON fields; (iii) the screenshots and event log are only aligned by ordering, without explicit timestamps, so correspondence between an image and nearby events must be inferred; and (iv) effective summarization requires domain-specific knowledge of Apex Legends mechanics and the ability to infer strategy from partial state observations.

Click here to view the sample data collection

4.1 Data preprocessing

We apply minimal preprocessing in order to preserve the realism of the raw telemetry while making it suitable for large language models.

First, we segment the raw recordings into 10-second collections and group the corresponding modalities into a unified structure: each collection directory contains the screenshots captured during that interval, the associated Overwolf event log snippet, and the matching video segment. We additionally record lightweight metadata (e.g., collection identifier and number of screenshots) to facilitate batching and analysis. No manual or algorithmic temporal alignment between log entries and frames is introduced; the only ordering constraint is the sequence in which events and screenshots were produced by the Overwolf API.

Second, we consider an *optional* normalization step for the textual event log. The raw log is emitted as nested JSON, and many fields are themselves JSON-encoded strings containing additional escape characters and structural noise. For some experiments, we directly provide this raw representation to the model after only basic cleaning (ensuring line-level validity and removing clearly spurious characters), thereby mimicking a realistic deployment scenario. In a separate, preprocessed variant, we repeatedly decode JSON-encoded strings, flatten nested structures, and extract a human-readable summary of salient fields (e.g., player identifiers, weapon names, damage values). This reduces token count and improves clarity without altering the underlying event sequence. Examples of the unparsed and parsed logs are provided in the project repository.[2]

We additionally explored using TOON (Token-Oriented Object Notation), a compact encoding format designed to reduce token consumption for LLM inputs while maintaining the JSON data model. TOON combines YAML-like indentation with CSV-style tabular layouts for uniform arrays, potentially offering further token reduction over standard JSON parsing. However, preliminary experiments showed that TOON encoding of our event logs provided minimal additional benefit beyond our existing JSON parsing approach, as our event logs were already relatively compact after flattening and extracting salient fields. For datasets with more uniform array structures, TOON might offer more substantial savings, but our semi-structured, event-driven logs did not match TOON's optimal use case of uniform arrays of objects.

Finally, we do not perform any preprocessing over the video beyond basic segmentation into 10-second clips. The video is not used as a model input in our current experiments; instead, it serves as a reference for human inspection and as a baseline modality for future work that reasons directly over raw video rather than structured logs and screenshots. Here are examples of unparsed vs parsed:

Click here to view full unparsed summary
Click here to view full parsed summary

5 Baselines

Our evaluation framework compares four different video summarization methods to demonstrate the efficiency gains of event-driven frame selection. The primary baseline is the raw video summarization method, which represents the traditional approach to LLM-based video analysis. This method sends the maximum number

[2] https://github.com/Kuzoto/Apex_AI/blob/main/sample_data/collection_20251119_165050/keyframes_log_unparsed.txt vs. https://github.com/Kuzoto/Apex_AI/blob/main/sample_data/collection_20251119_165050/keyframes_log_parsed.txt

of frames possible by dynamically extracting uniformly spaced frames and resizing them when necessary to stay within token limits. While this approach provides comprehensive visual coverage of the video content, it processes frames indiscriminately regardless of whether meaningful events are occurring, resulting in high token consumption and frequent hallucinations. This is the baseline we will be referring to in this report.

To establish that more intelligent frame selection methods can improve upon raw video processing, we implemented two intermediate baseline approaches using keyframe extraction. The first intermediate baseline, keyframes with unparsed event log, extracts the k most visually distinct frames from the video and sends them along with the event log as context. This method achieved an average of 6,284 tokens per video with 75.4% accuracy. The second intermediate baseline applies JSON parsing to the event log to remove extraneous characters and simplify formatting, reducing average token consumption to 5,861 tokens while maintaining 75.5% accuracy. These keyframe-based methods demonstrate that strategic frame selection reduces both token usage and hallucination frequency compared to raw video, while establishing a reliability benchmark that sits between traditional and optimal approaches.

We selected these baselines to create a clear progression demonstrating incremental improvements in efficiency and reliability. The raw video baseline establishes the traditional, resource-intensive approach currently used for LLM video analysis. The keyframe methods prove that intelligent frame selection offers meaningful improvements over indiscriminate frame processing, providing a more efficient and reliable middle ground. This staged comparison allows us to demonstrate that event-driven frame selection achieves optimal efficiency when structured event data is available, while the keyframe approach offers a practical fallback for scenarios with sparse event activity.

6 Your approach

Our approach demonstrates that event-driven context frame selection is promising for making video summarization by LLMs more efficient and accurate. We compare four different summarization methods to establish a progression from traditional to optimal frame selection strategies.

Table 1: raw_video Performance

Video ID	Total Tokens	Acc. (%)
Video 1	17,969	61
Video 2	17,969	64
Video 3	17,969	73
Video 4	17,793	69
Video 5	17,793	85
Video 6	17,912	48
Video 7	15,995	68
Video 8	14,934	79
Avg.	**17,292**	**68.4**

Method Overview The baseline *raw video summarization* method represents the traditional approach to LLM-based video analysis. This method sends the maximum number of frames possible by dynamically extracting uniformly spaced frames and resizing them when necessary to stay within token limits. While this provides comprehensive visual coverage, it processes all frames indiscriminately regardless of whether meaningful events are occurring.

Our primary contribution, *event-based frame summarization*, captures screenshots only when specific gameplay events occur. These triggering events include kills, knockdowns, assists, local match information updates (tab key presses), and teammate status changes. This method sends a complete log of all events that occurred in a 10-second interval along with screenshots captured at the exact moments of key events, enabling context-aware summarization with minimal visual input.

We implement two intermediate methods using keyframe extraction to demonstrate that reliable improvements exist even without optimal event-driven selection. The *keyframes with event log* methods extract the k most visually different frames from the video and send them alongside the event log as context. These methods establish a middle ground between comprehensive frame coverage and intelligent selection, demonstrating a more efficient and reliable approach for video summarization tasks.

Implementation Details All methods were evaluated using OpenAI's GPT-5 model. Video processing and frame extraction were implemented in Python using OpenCV (cv2), with event log parsing handled by custom scripts. The event-triggered screenshot capture system integrated with Overwolf's API through their JavaScript SDK. For keyframe extraction, we used the video-keyframe-detector library with 25 as the target frame count and 0.4 as the similarity threshold.

Results and Comparison When implementing the event-based frame summarization, we expected it to hallucinate similarly to the baseline and produce less detailed summaries as a result of having less visual information to process. Contrary to these expectations, hallucinations occurred far less frequently than anticipated, being very rare for this method. The detail of the summary did suffer in cases where few key events occurred. However, in cases where at least 10 screenshots were captured, the summaries were of similar detail to the baseline. The event-based summarization failed when little or no events occurred.

Table 2: event_frames_log Performance

Video ID	Total Tokens	Acc. (%)
Video 1	4,704	95
Video 2	4,704	100
Video 3	4,704	81
Video 4	4,342	69
Video 5	5,056	62
Video 6	4,342	53
Video 7	4,225	76
Video 8	5,151	89
Avg.	**4,654**	**78.1**

The event-based method achieved 78.1% accuracy with an average of 4,654 tokens per video—a 73% reduction in token usage compared to the raw video baseline's 17,292 tokens while improving accuracy by 14% (from 68.4% to 78.1%). This demonstrates that LLMs do not need to see every frame equally; structured event data effectively guides the model to focus on relevant moments.

Fallback Strategy In practice, for real-time summarization, we implemented a fallback mechanism that utilizes the keyframe method when few event frames are captured. This still drastically reduced token usage compared to the baseline while keeping accuracy relatively high. The keyframe method is meant to demonstrate that there exists a more efficient and reliable method for video summarization, though it is not the most efficient approach. This method used more tokens than the event-based method but kept accuracy very similar. It also resulted in fewer hallucinations compared to the raw video baseline.

Table 3: keyframes_log_unparsed Performance

Video ID	Total Tokens	Acc. (%)
Video 1	5,574	74
Video 2	5,574	93
Video 3	5,574	79
Video 4	7,203	77
Video 5	6,236	65
Video 6	7,203	56
Video 7	6,745	77
Video 8	6,163	82
Avg.	**6,284**	**75.4**

Token Optimization To test further token reduction and improved event-log clarity, we evaluated JSON parsing on the keyframe method to remove extra characters and simplify log formatting. This preprocessing decreased the average token consumption for the keyframe method by 400 tokens, but had little effect on accuracy, demonstrating that removing syntactic noise from event logs preserves semantic content while reducing costs.

Table 4: keyframes_log_parsed Performance

Video ID	Total Tokens	Acc. (%)
Video 1	5,033	64
Video 2	5,033	79
Video 3	5,033	62
Video 4	7,158	96
Video 5	5,427	73
Video 6	7,158	75
Video 7	6,072	78
Video 8	5,975	76
Avg.	**5,861**	**75.5**

Evaluation Methodology Most of the accuracy evaluations, each video excluding video 1, were performed using Claude Sonnet 4.5 as an LLM-as-a-Judge. For each of the 7 videos, Claude was given the summaries produced using each method, the video they were summarizing, and the event log associated with the video. Before the evaluation prompts, Claude was given the following instruction:

In the following prompts, I will be feeding you 4 summary files all summarizing the same video but using different contents. Additionally, I will also provide the log file used in some of the summaries and the video. I want you to produce a evaluation

file similar to the example provided in eval.txt using the summary and log files and videos provided in future prompts. Here are the summary files used to create eval.txt and the video and log file they were generated using.
A link to the chat log for the LLM-as-a-Judge evaluations is included in Section 2.

The accuracy of the summary was determined by counting the total number of statements and the number of incorrect statements, then a percentage was acquired using the formula:

$$Accuracy = \frac{statements - incorrect}{statements} \times 100\%$$

This metric focuses on factual correctness—statements represent factual claims made in the summary, and incorrect represents claims contradicted by the video or event log. Further evaluation into event coverage, measuring how much of the video was addressed, may be conducted in future work. For the sake of this experiment, we chose to focus on summary accuracy as the primary quality metric.

7 Error analysis

Through systematic evaluation of 8 gameplay videos, we conducted a comprehensive manual error analysis covering 164 incorrect statements across approximately 200 factual claims. This analysis reveals distinct failure patterns for each method, with the raw_video baseline producing 58 errors (35.2% error rate), our event_frames_log approach producing 29 errors (22% error rate), and the intermediate keyframes methods falling between these extremes. The failures cluster into predictable categories based on systematic weaknesses in visual grounding, numerical precision, temporal reasoning, and narrative generation.

The most pervasive failure mode across all methods is character and weapon misidentification. In 5 out of 8 videos, methods incorrectly identified the player character, claiming the player was Loba, Seer, Octane, Mirage, or Ballistic when in all clips they were actually playing Sparrow. Similarly, the R-99 SMG was misidentified as the R-301 Carbine in 6 of 8 videos. This confusion appears directional—methods almost never misidentify an R-301 as an R-99, suggesting they default to reporting the more common weapon when uncertain. We hypothesize this occurs because the models rely on global appearance features (silhouette, color palette) rather than discriminative local details (weapon attachments, iron sights). Resolution limitations compound this issue—when videos are downsampled for model input, fine details that distinguish similar weapons are lost, forcing models to guess based on coarser visual features shared across weapon types.

Hallucinated game elements represent the most severe failure category. Video 3 showed keyframes_log_parsed inventing an entire Phase Runner portal traversal scenario that never occurred. Video 5 saw multiple methods claiming a ”vertical blue energy column” appeared near an enemy, which upon inspection was likely bullet tracer effects misinterpreted as persistent objects at lower resolutions. Video 6 showed keyframes_log_unparsed repeatedly mentioning a Bocek Bow weapon throughout the summary, building an entire tactical narrative around a weapon that never appeared. These hallucinations share a common pattern: they are plausible game elements that commonly appear in Apex Legends. We hypothesize a two-stage failure mechanism where models first detect genuine patterns (defensive positioning, chokepoint control), then retrieve associated elements from training data (Wattson fences, Gibraltar domes) and generate these as observations. The generation process lacks verification mechanisms to check whether retrieved elements are actually present versus merely plausible. Resolution limitations worsen this by blurring distinctions between transient effects (bullet trails lasting a few frames) and persistent objects.

Statistical and numerical processing failures affected all methods consistently. Methods reported incorrect EVO values (claiming ”52 EVO” when the actual value was 524), misreported squad counts (claiming 15 squads when only 3 were shown), and incorrectly stated kill counts. Numbers were frequently transposed, rounded, or approximated, with particular difficulty on 3-digit numbers where models often dropped the hundreds place (524 becomes 24 or 52). We hypothesize this stems from two sources: first, general vision-language models are not optimized for OCR tasks on game HUDs, which have different visual characteristics than natural text; second, resolution limitations create a fundamental information bottleneck. If HUD text appears at 12-16 pixels tall in processed frames, even human observers would struggle to read it accurately. Teammate name spelling errors (ShadyDinoXL vs ShadyDrexXD) follow the same pattern—models preserve name structure (prefix, suffix) but corrupt middle segments, suggesting they use contextual priors to fill in details they cannot clearly see.

Our event_frames_log approach exhibits qualitatively different failures than the baseline. The most common failure is incomplete teammate status tracking—in Videos 4, 6, 7, and 8, the method failed to recognize teammates in "RESPAWNING" state or needing banner delivery, instead claiming "full squad." This occurs because log data reports binary alive/dead status, while visual indicators like "DELIVER TO BEACON" appear only in the HUD. When event frames are selected based on combat events (damage, kills), these status indicators may not be captured if they fall between selected frames. We hypothesize methods overly rely on high-salience events for frame selection, causing them to miss lower-salience state indicators. Additionally, temporal misalignment between log timestamps and frame extraction accounts for approximately 25% of errors—damage numbers "correct" according to logs don't match visible HUD values when timestamps misalign.

The intermediate keyframes methods show that increased visual sampling without careful frame selection enables compound errors. keyframes_log_unparsed in Video 6 generated sustained Bocek Bow hallucinations across multiple sentences, maintaining narrative consistency around a weapon that never appeared. We hypothesize that autoregressive generation creates commitment effects where early errors constrain later generation. Once a model generates "Bocek" in one sentence, maintaining consistency drives it to continue referencing the Bocek, even as new frames contradict this interpretation. The model lacks mechanisms to revise earlier claims based on later evidence. This suggests better architectures might require bidirectional reasoning or explicit revision steps.

Analyzing error patterns reveals several cross-cutting commonalities. First, visual ambiguity under motion and occlusion—errors cluster around weapon swaps, ability activations, and camera transitions where multiple interpretations are plausible. Second, default to common patterns—when uncertain, all methods bias toward reporting common gameplay elements (R-301 over R-99, 15 squads over 3) rather than correct but less common observations. Third, category confusion within semantic clusters—errors rarely cross major boundaries (weapon as grenade) but frequently occur within categories (SMG confused with another SMG). Fourth, narrative coherence over grounding—the most severe hallucinations occur when models prioritize maintaining coherent narratives over strict visual accuracy. These patterns suggest models are pattern-matching against learned gameplay scripts rather than purely describing visual content.

Our event_frames_log approach substantially reduces failures from 35.2% to 22% error rate by grounding summaries in log data, but introduces new failure modes related to frame selection timing and data stream synchronization. The baseline raw_video method fails primarily through hallucination and visual ambiguity, while our approach trades these failures for occasional status tracking misses and temporal misalignment issues. Resolution limitations affect all methods, creating fundamental information bottlenecks for text reading, fine-grained discrimination, and transient effect interpretation. Addressing these failures requires not just better models but improvements in multi-modal grounding, temporal reasoning, verification mechanisms, and maintaining higher resolution for HUD elements throughout the video processing pipeline.

8 Contributions of group members

List what each member of the group contributed to this project here. For example:

- Christian Calvo: performed data collection, method implementation, and evaluation
- Sunggyeol (Sung) Oh: Implemented video key frame extraction, JSON token reduction, and wrote related work
- Ermias Asnake: Evaluation, Conclusion
- Junwoo Seo: Evaluation, Data Analysis
- Siddharth Rakshit: Evaluation, Future Works

9 Future Work

Our research demonstrates that event-driven frame selection significantly improves the token efficiency of video summarization. However, our current reliance on the Overwolf API limits the system's applicability to games with structured external APIs. To address this, we propose the following directions for future research.

9.1 Training a Vision Model

The primary limitation of our current approach is its dependency on the Overwolf API for event detection. Future work should focus on developing a standalone computer vision model capable of extracting event frames and event lists directly from raw video content.

The multimodal dataset collected in this project (time-synced screenshots, event logs, and raw video) provides an ideal ground-truth training set. A Convolutional Neural Network (CNN) or a fine-tuned Vision Transformer (ViT) could be trained on this data to recognize visual cues corresponding to game events. The objective is to create a model that outputs a structured event list and keyframe timestamps solely from video input. This would allow the "Event Frame Summary" method to be applied to any video game or video domain without requiring game-specific API integration.

9.2 Applying Reinforcement Learning

While our current "Event Frame" method relies on a static set of heuristic rules (e.g., "always capture on Kill"), this rigid logic may not be optimal for every scenario. We propose implementing a Reinforcement Learning (RL) framework to dynamically learn the optimal frame selection policy. An RL agent could be trained to select frames with the goal of maximizing a composite reward function. This function would penalize high token usage while rewarding high summary accuracy (as measured by an LLM-as-a-Judge). By utilizing Reinforcement Learning from Human Feedback (RLHF) or AI feedback, the model can learn to identify non-obvious context frames that static rules might miss - such as capturing the moments leading up to a fight rather than just the kill itself - thereby reducing the "missing context" errors observed in our results.

10 Conclusion

One key takeaway from our project was that intelligent frame selection based on structured event data fundamentally improves video summarization efficiency. Our event_frame method had the best performance on paper, 78.1% accuracy with only 4654 tokens, making it 73% cheaper than the raw_video method with 14% better accuracy. This shows that LLMs don't need to see every frame equally. Another key takeaway was that methods with log data outperformed video-only analysis by 8-12% accuracy while using significantly few tokens. This shows that external structured context act as a useful guide for visual understanding, helping the LLM focus on what matters rather than the noise.

One thing that comes to mind that was difficult was low event scenarios. Even though it was the most token efficient, event_frames often missed on key context when a few key events happened. This showed a very important dual-edged sword: maximum efficiency required sufficient event density. Falling back to keyframe extraction reduced the method's elegance but increased its robustness. The persistence of common errors across all methods was interesting. Legend misidentification, weapon confusion (R-301 vs R-99), and hallucinated elements (Gibraltar domes, Watson fences) appeared regardless of approach. This entailed that the errors come from fundamental limitations in vision language models rather than our frame selection strategy. Misalignment between Overwolf API timestamps and actual HUD values created errors; about 25% of mistakes involved numbers being correct in logs but wrong visually. This showed the difficulty of synchronizing multiple data streams in real-time systems.

We weren't surprised too much by the results, but there were a few things. One thing we thought was that event-based summarization was going to hallucinate more due to having less data, but the hallucinations weren't common when there was at least 10 event screenshots captured. The structured event grounded the LLMs' responses more effectively than just providing more frames, showing that quality mattered more than quantity. Another thing was how drastic the token reduction was without accuracy loss. Methods using log data had 75-78% accuracy at around 4600-6300 token, while raw_video had only 68.4% accuracy at 17292 tokens. This was just a significant improvement; it showed that traditional frame-by-frame analysis wastes tokens on useless information. JSON parsing for token reduction also had barely any effect on accuracy, 75.4% vs 75.5%, but was able to save at least 400 tokens. This however showed that removing the extra characters from event logs kept all the meaningful information while still reducing costs.

Table 5: Summary Statistics by Method

Method	Avg. Tokens	Avg. Acc%
event_frames	4,654	78.1
kf_parsed	5,861	75.5
kf_unparsed	6,284	75.4
raw_video	17,292	68.4

Table 6: Token Usage Breakdown by Method

Method	Prompt	Compl.	Total
event_frames	664	3,990	4,654
kf_parsed	1,036	4,825	5,861
kf_unparsed	1,033	5,251	6,284
raw_video	12,280	5,012	17,292

Table 7: Cost Efficiency Analysis

Method	Tokens/ Acc%	Rank
event_frames	59.6	1st
kf_parsed	77.6	2nd
kf_unparsed	83.3	3rd
raw_video	252.8	4th

11 AI Disclosure

- Did you use any AI assistance to complete this proposal? If so, please also specify what AI you used.
  - Yes, Claude Sonnet 4.5, Gemini 2.5 Pro

If you answered yes to the above question, please complete the following as well:

- If you used an AI to assist you, please paste *all* of the prompts that you used below. Add a separate bullet for each prompt, and specify which part of the proposal is associated with which prompt.
  - Error analysis: can you use the summary method analysis to answer the following questions in paragraph form?
    What kinds of inputs do your baselines fail at? What about your approach? Are there any semantic or syntactic commonalities between these difficult examples? **We would like to see a manual error analysis (e.g., annotate around 100 (20 * group_size) or more failed examples for various properties, and then discuss the results and hypothesize about why the model fails in each category).**
  - Error analysis: mention that the confusion of energy columns could have been referring to bullet trails and that name and weapon confusion could be caused by low resolution making it hard to distinguish names since incorrect text analysis was also observed for evo progress. These paragraphs should all lie in one section of a report.
  - can you make the style and wording similar to the style of the following text? Also can you shorten it to max 6-8 paragraphs, focusing on hypothesis and error types?
    PASTED: Approach
  - Tables: Can you create a table for each method with their token usage and accuracy per video and their averages for each at the end?
    I want to paste this into latex and google slides
  - For the conclusion, summarize everything and list the main takeaways and the most surprising things on the difficulties and results of the project
    Conclusion
  - Related work: Help me summarize the research papers I found
  - Dataset: Help me explain TOON (Token-Oriented Object Notation) before the final paragraph
  - All parts: Please give me a list of paragraphs that need to be more elaborated or added to meet the criteria in the template. I attached both the template and our draft, as well as a PDF version of the draft.

- **Free response:** For each section or paragraph for which you used assistance, describe your overall experience with the AI. How helpful was it? Did it just directly give you a good output, or did you have to edit it? Was its output ever obviously wrong or irrelevant? Did you use it to generate new text, check your own ideas, or rewrite text?

– Error analysis: It was fairly helpful, I needed to ask it to shorten the output and correct some incorrect analysis but it was mostly right. I used it to perform the error analysis since the LLM-as-a-Judge performed a lot of the evaluations.
– Tables: It was very helpful, it generated comparison tables for each method very quickly and I hardly needed to edit it.
– Conclusion: It was helpful. It gave me a good output first try. I used it to rewrite text.
– Related work, Dataset: It was helpful to elaborate better on what I currently have.
– All parts: It was helpful to revise my draft overall.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew S Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikołaj Bińkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. 2022. Flamingo: a visual language model for few-shot learning. *36th Conference on Neural Information Processing Systems (NeurIPS 2022)*.

Guo Chen, Yin-Dong Zheng, Jiahao Wang, Jilan Xu, Yifei Huang, Junting Pan, Yi Wang, Yali Wang, Yu Qiao, Tong Lu, and Limin Wang. 2023. Videollm: Modeling video sequence with large language models.

Joya Chen, Zhaoyang Lv, Shiwei Wu, Kevin Qinghong Lin, Chenan Song, Difei Gao, Jia-Wei Liu, Ziteng Gao, Dongxing Mao, and Mike Zheng Shou. 2024. Videollm-online: Online video large language model for streaming video.

Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. 2023. Instructblip: Towards general-purpose vision-language models with instruction tuning.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. 2021. An image is worth 16x16 words: Transformers for image recognition at scale.

Difei Gao, Luowei Zhou, Lei Ji, Linchao Zhu, Yi Yang, and Mike Zheng Shou. 2023. Mist : Multi-modal iterative spatial-temporal transformer for long-form video question answering. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14773–14783.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven . 2022. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. *Proceedings of the 39th International Conference on Machine Learning (PMLR)*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning.

Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*.

Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. 2023. Video-chatgpt: Towards detailed video understanding via large vision and language models. *Association for Computational Linguistics*.

Rui Qian, Xiaoyi Dong, Pan Zhang, Yuhang Zang, Shuangrui Ding, Dahua Lin, and Jiaqi Wang. 2024. Streaming long video understanding with large language models. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA. Curran Associates Inc.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. *Preprint*, arXiv:2103.00020.

Yuetian Weng, Mingfei Han, Haoyu He, Xiaojun Chang, and Bohan Zhuang. 2024. Longvlm: Efficient long video understanding via large language models. *ECCV 2024: 18th European Conference*, pages 453–470.