GlitchAgent: Detecting Video Game Glitches from Gameplay Videos

Tong Zhou

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Science and Applications

Lifu Huang, Chair
Xuan Wang
Christopher L Thomas

August 15, 2025
Blacksburg, Virginia

Keywords: Multimodal Large Language Model, Video Understanding, Glitch Detection.

GlitchAgent: Detecting Video Game Glitches from Gameplay Videos

Tong Zhou

(ABSTRACT)

The increasing complexity of modern video games has made Quality Assurance (QA) a critical yet challenging bottleneck in the video game development and maintenance lifecycle, which relies heavily on expensive, labor-intensive, and inefficient manual testing. Automated glitch detection from gameplay videos offers a promising alternative, but is hampered by a profound scarcity of annotated datasets, the ambiguity of identifying glitches without temporal context, and the need for precise temporal localization of anomalies. In this thesis, we propose a novel approach to address these challenges. First, we introduce a new video-based benchmark dataset VideoGlitch for video game glitch detection, featuring diverse gameplay videos. The videos are annotated with detailed, natural-language glitch descriptions and precise temporal timestamps, created through a semi-automated pipeline leveraging Multimodal Large Language Models (MLLMs) and human validation. Second, we propose GlitchAgent, a multi-stage framework for open-ended glitch detection with precise timestamps. GlitchAgent operates by different video preprocessing procedure, then generating glitch hypotheses with the Local Glitch Detector, tracing the full duration of anomalies via a novel temporal propagation mechanism, and synthesizing a single, temporal description for each unique glitch with corresponding timestamps. To evaluate our system, we introduce the LLM-as-the-judge Glitch Detection Score (GDS), a novel metric that uses an LLM for semantic scoring and couples it with temporal Intersection over Union (IoU) for a more robust assessment than traditional metrics. Experiments demonstrate that GlitchAgent significantly enhances the

performance of various MLLM backbones, substantially improving detection precision and temporal grounding accuracy compared to baseline approaches.

GlitchAgent: Detecting Video Game Glitches from Gameplay Videos

Tong Zhou

(GENERAL AUDIENCE ABSTRACT)

Video games are more complex and immersive than ever, but this complexity often leads to frustrating "glitches" or bugs—like characters getting stuck in walls, objects behaving strangely, or quests that can't be completed. Traditionally, finding these glitches relies on huge teams of human testers who play the game over and over, a process that is slow, expensive, and can't possibly catch every error in a massive game world. This thesis introduces GlitchAgent, an artificial intelligence (AI) system designed to automatically find and report these glitches by watching videos of the game being played. Think of GlitchAgent as a tireless, superhuman game tester. It breaks down long gameplay videos into small, manageable clips, and the AI meticulously examines each clip for anything unusual. When it finds a potential glitch, it then intelligently tracks the problem backward and forward in time to pinpoint exactly when the glitch started and when it ended. Finally, it writes a clear, concise summary of the problem—for example, "The main character's horse started floating in the air from the 1-minute, 15-second mark to the 1-minute, 22-second mark"—creating an automated bug report for developers to fix. To build and test this AI, we first created the largest-ever public library of video game glitches, complete with detailed descriptions and exact timings. Our experiments show that the GlitchAgent system is effective at identifying and describing glitches with high accuracy. The goal of this research is to help game developers create more stable, polished games more efficiently, reducing development costs and leading to a better experience for players.

Dedication

Dedicated to my mom, thank you for your constant support and companionship.

Acknowledgments

I am deeply grateful for the careful guidance of my advisor, Lifu Huang. I would also like to thank Jingyuan Qi and Zhiyang Xu for their invaluable help and mentorship. I have learned a great deal throughout the research process.

Contents

List of Figures

List of Tables

List of Abbreviations

f_s	Sampling rate
g	Generated report
gt	Ground-truth report
S_{LLM}	LLM similarity score
T_p	Predicted time ranges
T_{gt}	Ground-truth time ranges
w_s	Window size
AAA	High-Budget (as in AAA titles)
AI	Artificial Intelligence
API	Application Programming Interface
COCO	Common Objects in Context
F1	F-measure Score
FPS	Frames Per Second
GDS	LLM-Judged Glitch Detection Score
ID	Identifier
IoU	Intersection over Union

JSON JavaScript Object Notation

LLaVA Large Language and Vision Assistant

LLM Large Language Model

MCQA Multiple-Choice Question Answering

METEOR Metric for Evaluation of Translation with Explicit ORdering

mIoU mean Intersection over Union

MLLM Multimodal Large Language Model

MMORPG Massively Multiplayer Online Role-Playing Game

MoE Mixture-of-Experts

NPC Non-Player Character

QA Quality Assurance

RPG Role-Playing Game

RTT Real-time Tactics

SODA Story-Oriented Dense video cAptioning evaluation framework

VideoMAE Video Masked Autoencoders

VLM Vision-Language Model

Chapter 1

Introduction

The video game industry has become a global entertainment titan with revenues that outstrip those of the film and music industries put together. Modern AAA titles are no longer simple applications but have grown into massive, complex software ecosystems. These titles feature high-fidelity graphics, massive open worlds, advanced physics engines, dynamic AI, and sprawling, interactive narratives. Games like Cyberpunk 2077 or Red Dead Redemption 2 contain millions of lines of code and hundreds of gigabytes of art, audio, and logic assets.

However, this very complexity that fosters so much immersion is also accompanied by a commensurate increase in the potential for error. In almost all cases, these are tightly coupled systems, where graphics, physics, AI, and game logic have to work in unison; such a structure breeds defects. Hence, QA has matured into a pillar of the game development and maintenance process and has become a critical factor in delivering a stable and enjoyable experience to market.

1.1 Background and Motivation

The predominant method for QA has remained manual testing. This process involves employing large teams of human testers to play the game repeatedly, trying to explore every possible state, interaction, and corner of the game world. This method is highly labor-intensive, time-consuming, and expensive. Developers often spend millions of dollars and

thousands of person-hours on QA alone. This dependence on manual labor contributes to the so-called "crunch culture," in which teams must work extensive overtime to meet deadlines.

Beyond that, manual testing is limited. It is a gargantuan task to test every possibility a big game might offer. A subtle bug misses a human tester's eye, or they cannot replicate the bug's occurrence consistently. In other cases, the tester may simply become tired and thus less effective. Humans excel at pinpointing context-dependent and unexpected issues—tasks where machines typically struggle. However, comprehensive testing that requires endless repetition is a task ill-suited for humans.

Hence, an automated solution is required. We address detecting glitches on gameplay videos in this study. The player provides recorded videos containing instances of game glitches, and the proposed system automatically detects these glitches (what and when) within the videos.

1.1.1 Taxonomy of Glitch

In the context of this research, a "glitch" is defined as any observable behavior within the game that deviates from the intended design and functionality, negatively impacting the player's experience. These can range from minor aesthetic annoyances to game-breaking bugs that halt progression entirely. We do not group all glitches into rigid categories, as they can be persistent, mixed, and arise from various sources, which could mislead developers. The following types are often perceived as a glitch in the experience.

- **Visual Glitches**: These are basically defects regarding the graphical rendering of the game whose examples include texture popping (where the low-res texture fails to get replaced with a high-res texture), T-posing (when a character reverts to a default

animation pose), incorrect lighting, and shader artifacts.

- **Physics Glitches**: These are errors arising from the game's physics engine. Common examples include ragdoll physics acting erratically, objects or characters falling through map geometry, sudden explosions that fling things into the air, and model clipping where portions of a character model passes through another object.

- **Game Logic Glitches**: These are glitches in the core rules and systems of a game. These could comprise quests that cannot be finished, NPCs refusing to respond or getting stuck in an animation loop, items that do not work as they are described, or even exploits that unfairly advantage a player.

- **Other Glitches**: Those that concern the performance of the game, such as stuttering, sudden frame rate drops, or freezing; lag on the internet; and camera issues- all of which detract from the player experience.

1.1.2 Challenges of Detecting Glitches from Videos

The Scarcity of Annotated Glitch Data A primary obstacle is the profound scarcity of large-scale, high-quality, and richly annotated datasets for video game glitches. Unlike established computer vision domains like object detection or action recognition, which benefit from massive public datasets (e.g., COCO, Kinetics), no such resource exists for game anomalies. Glitches are, by nature, "long-tail" events—unexpected and often rare edge cases. Compiling a diverse and comprehensive corpus requires an immense amount of gameplay footage. Furthermore, annotation is a labor-intensive process that demands more than just a label. A useful dataset would require:

- **Video clips** showcasing a wide variety of glitches across different games and genres.

- **Precise temporal annotations** (start and end times) for each glitch event.
- **Detailed, natural language descriptions** of what is occurring in the glitch.

Game development studios possess this data internally, but it is proprietary, often tied to unreleased intellectual property, and not formatted for academic use. This data bottleneck severely constrains the training and, just as importantly, the robust evaluation of supervised models for this task.

Task Formulation: From Simplistic to Practical The very formulation of the glitch detection task is non-trivial and has been a limitation in prior work. Simply treating glitch detection as a static image analysis problem is fundamentally flawed. Motion and temporal context are essential for distinguishing between intended game mechanics and true bugs. For instance, a character suspended in mid-air within a single frame is ambiguous; they could be performing a legitimate jump, using a flight ability, or be stuck in a physics glitch. Only by observing the sequence of events in a video can a correct determination be made.

In addition, a few studies have attempted to make the problem more tractable by reframing it in the form of a multiple-choice question answering problem, where a model selects a pre-written description that matches the glitch in a video clip. While useful for benchmarking model comprehension, this approach does not mirror a real-world testing scenario. In practical game development, a QA system receives no hints or predefined options. It must identify and describe anomalies from raw observation alone. Therefore, a more practical and challenging task formulation—and the one adopted in this thesis—requires a model to perform **open-ended glitch detection**. The system must autonomously determine that an anomaly has occurred and generate a novel, descriptive bug report from scratch, akin to the process performed by a human tester.

The Challenge of Precise Temporal Grounding Building upon the need for temporal context, a critical and often overlooked requirement is precise **temporal grounding**. For a generated bug report to be actionable for a developer, it is not sufficient to simply flag an entire 60-second video clip as "containing a glitch." A developer needs to know precisely when the anomaly occurred to effectively debug the issue.

This elevates the task from a mere video classification task (glitch/no-glitch) into a highly complex localization problem. Ideally, the system should identify and describe the glitch event while simultaneously providing the exact start and end timestamps of the event within the overall video stream. This merger of tasks constitutes a multi-task problem: from detecting whether or not a glitch occurs, describing what it is, and localizing it within precise time boundaries, it requires a fine-grained understanding of the video's timeline and making it far more sophisticated than each of these tasks taken in isolation, overstretching the boundaries of current video-understanding capabilities.

1.2 Proposed Solution

The thesis offers a solution that addresses the previously mentioned challenges of scarce data, open-ended task formulation, and the need for a finer temporal grounding. Three contributions form the basis for our approach: the benchmark dataset **VideoGlitch**, the novel detection framework entitled **GlitchAgent**, and the more rigorous evaluation metric **LLM-Judged Glitch Detection Score (GDS)**.

First and foremost, to address the profound scarcity of data, we introduce **VideoGlitch**, a novel benchmark for video game glitch detection.. It is carefully kept diverse, with a range of games, genres, and types of glitches. Crucially, videos are annotated-with a detailed natural language description of the glitch, along with the exact start and end time of its occurrence.

Rich annotations come into existence through a new semi-automated pipeline that gives initial pseudo-descriptions by exploiting the video understanding capabilities of the MLLMs; these pseudo-descriptions are then refined and verified by human annotators.

Second, to move beyond simplistic classification and address the practical need for **open-ended detection and temporal grounding**, we propose **GlitchAgent**, a multi-stage framework for automated glitch analysis. Rather than processing a video in its entirety, GlitchAgent operates as follows:

1. **Window-based Hypothesis Generation:** The framework first segments a long video into a series of short, non-overlapping windows. An MLLM analyzes each window independently to generate initial "glitch hypotheses," maximizing sensitivity to brief or subtle anomalies.

2. **Temporal Propagation:** Verified glitch hypotheses act as "anchors." A novel temporal propagation mechanism then traces the full duration of a glitch by systematically searching adjacent windows—both backward and forward in time—for the same specific anomaly. This process precisely localizes the full start-to-end timeline of the event.

3. **Canonical Description Generation:** Finally, for each unique glitch instance identified and tracked, the MLLM synthesizes all related textual descriptions into a single, cohesive, and definitive canonical report, eliminating redundancy and creating an actionable summary.

Finally, to overcome the limitations of existing evaluation protocols, we introduce the **GDS**. Traditional metrics like SODA that leverages METEOR score, which rely on lexical overlap, fail to capture the semantic nuance of glitch descriptions and match the most similar glitch

description pairs. Our GDS metric replaces this with an **LLM-as-judge** for semantic scoring, which is then coupled with the temporal Intersection over Union (IoU). This method provides a far more robust and human-aligned assessment of system performance, rewarding both descriptive accuracy and temporal precision.

1.3 Thesis Structure

This thesis is organized into the following chapters:

Chapter 1: Introduction introduces the significance of Quality Assurance in the modern video game industry, details the high costs and limitations of manual testing, and formally defines the core challenges this research addresses: data scarcity, practical task formulation, and precise temporal grounding. It concludes with an overview of our proposed contributions.

Chapter 2: Review of Literature surveys existing work relevant to our research. It first a review of available datasets for video game glitch detection, highlighting their limitations. It then discusses recent advancements in Multimodal Large Language Models (MLLMs), focusing on their application to video understanding and the challenges they face with long-form video and anomaly detection.

Chapter 3: VideoGlitch Dataset Collection provides a detailed account of the creation of our novel benchmark dataset. This chapter elaborates on the video selection criteria, the semi-automated pipeline for generating pseudo-descriptions using MLLMs, and the human validation and temporal annotation process.

Chapter 4: Methodology presents the technical architecture of our proposed framework, **GlitchAgent**. It details the multi-stage process, including video preprocessing, window-based glitch hypothesis generation, the temporal propagation mechanism for full-duration

tracking, and the final canonical description generation and merging stage.

Chapter 5: Experiments describes the experimental setup, introduces our novel evaluation metric, the Glitch Detection Score (GDS), and presents the results. It includes a comprehensive comparison of GlitchAgent's performance against various baseline MLLMs and provides detailed ablation studies to analyze the impact of different components and hyperparameters, such as sampling rate and window size.

Chapter 6: Conclusions summarizes the key findings of this thesis, discusses the implications of our work for automated game testing, acknowledges its limitations, and suggests potential directions for future research.

Appendices provide supplementary materials, including detailed statistics of our dataset and the specific prompts used to guide the MLLMs in our experiments.

Chapter 2

Review of Literature

2.1 Vision Glitch Detection Datasets

GamePhysics [16] is a large gameplay video recording collection consisting of 26,954 videos in 1,873 different video games collected from the GamePhysics subreddit[1], which includes thousands of the game community reports. GameBugDescription [17] is a benchmark dataset constructed based on GamePhysics, which consists of 167 buggy gameplay videos and a total of 334 question-answer pairs across 8 games. However, the descriptions within this benchmark are notably concise and exclusively crafted by human annotators. Additionally, GlitchBench [18] is another benchmark annotated based on GamePhysics with 593 images of video game glitches from 205 games. However, its problem is also the overly brief glitch description, and it judges the glitch based solely on one image, which is insufficient for glitch detection. PhysGame [5] is a benchmark dataset consisting of 880 gameplay videos containing glitches, each annotated with a multiple-choice question answer pair specifically addressing the nature of the glitch.

The detailed comparison of existing datasets is in Table 2.1.

Since existing datasets do not adequately address these challenges, we are motivated to develop a new benchmark. It contains many gameplay videos spread across various game titles,

[1] https://www.reddit.com/r/GamePhysics/

Dataset	Pipeline	Annotation	Timestamp
GamePhysics [16]	No	No	✗
GameBugDescription [17]	Manual	MCQA	✗
GlitchBench [18]	Manual	Simple description (Image)	✗
PhysGame [5]	Semi-Automated	MCQA	✗
VideoGlitch (Ours)	Semi-Automated	High-quality description	✓

Table 2.1: Comparison of Video Game Glitch Datasets

genres, and styles, thereby depicting a broad spectrum of gameplay experiences. Detailed natural language descriptions of the glitches with precise timestamps indicating when those glitches occur accompany this.

2.2 Multimodal Large Language Models for Video Understanding

This research builds upon recent and rapid developments in MLLMs. Generally, MLLMs bridge the divide between perception and reasoning with a pre-trained Vision Encoder and a pre-trained LLM. Architectures such as Flamingo [2], BLIP-2 [10], and LLaVA [11, 12] had shown great success on static images with detailed visual conversations and following comments about the image content.

Extending these capabilities from static images to dynamic videos introduces the critical challenge of modeling the temporal dimension. Early and straightforward approaches treated video as a "bag of frames," sampling a few keyframes and feeding them to an image-based MLLM. While simple to implement, this method loses essential motion and causality information, making it unsuitable for tasks where temporal context is key—such as distinguishing a jump from a levitation glitch.

To address this, subsequent works have focused explicitly on integrating temporal awareness into the MLLM architecture. Models like Video-LLaMA [22] and Video-ChatGPT [13] propose architectures that incorporate video-specific modules. These models often employ strategies such as:

- **Temporal Feature Aggregation**: It mainly involves the use of a pre-trained video encoder (e.g., VideoMAE [20]) to extract spatiotemporal features from video clips. Thereupon, the features are aggregated or pooled and projected into the LLM embedding space.
- **Instruction Tuning on Video-Text Data**: These models undergo a fine-tuning process on large-scale video-text datasets, where they learn to associate linguistic descriptions with visual events occurring in unison over time. They are trained for video summarization, video captioning, and video-question answering.

Such works have been used to demonstrate that MLLMs have the ability to achieve an advanced conception of the more common actions and events in the video. However, they are typically trained and tested on datasets consisting of conventional human activities (e.g., cooking, sports). Hence, the investigation of their applicability to detecting abnormal, out-of-distribution events such as video game glitches—anomalies that inherently carry no pre-established "action class"-is still largely unexplored.

However, they also share a common issue, which is the limitation on input length and the performance degradation with longer inputs. The method we propose can handle videos of any length and achieves performance improvements by obtaining more detailed information.

Chapter 3

VideoGlitch Dataset Collection

In this chapter, we elaborate on the pipeline for data collection and processing of this dataset VideoGlitch, as shown in Figure 3.1. In Sec.3.1, we explain the sources of our data and the criteria for video and game selection according to the genres of the game. In Sec.3.2, we construct a generation pipeline by leveraging powerful MLLMs to obtain pseudo glitch descriptions. In Sec.3.3, we explain the human corrections to pseudo descriptions and the annotation of glitches' temporal information.

3.1 Selection of Source Videos and Games

Our dataset originates from GamePhysics. To ensure the diversity of the dataset we construct, we first establish a set of game classifications based on the definitions of game genres

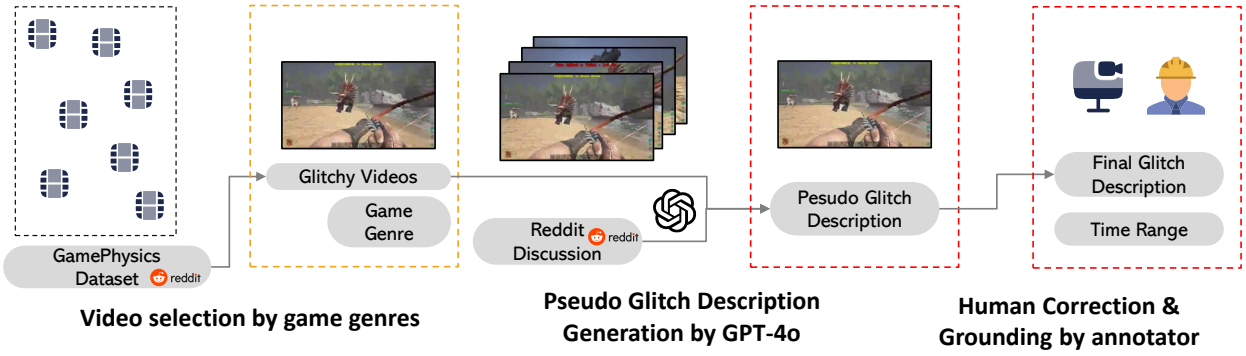


Figure 3.1: Data Collection Pipeline

based on Wikipedia[1], as shown in Table 3.1.

Game Genre	Sub-genres
Action	Action Adventure, Stealth, Fighting, Battle Royale, Platform, Survival, Shooter, Rhythm, Wargame
Simulation	Sports, Construction and Management, Vehicle Simulation, Life Simulation
RPG	Action RPG, Dungeon Crawl, Roguelike, MMORPG
Adventure	Interactive Film
Puzzle	
Strategy	Real-time Tactics (RTT), Turn-based Strategy

Table 3.1: Game Genres and Sub-genres

For all 1,873 different games in GamePhysics, we use gpt-4o-mini to categorize the games into predefined game genres and sub-genres. In GamePhysics, a significant number of games have very few videos, with many having only one or two. As a result, we discard games with fewer than ten videos, leaving us with a total of 364 games.

The distribution of game genres is quite uneven; some sub-genres have only one game that fits, such as the `Interactive Film` sub-genre within the `Adventure` genre, which includes only `Detroit: Become Human`. Therefore, we prioritize selecting videos from sub-genres and games with fewer entries to ensure their presence in the dataset. Detailed selection results are included in Appendix A.1.

[1] https://en.wikipedia.org/wiki/Action_game

3.2 Automated Generation of Pseudo Glitch Descriptions

Given the increasingly powerful performance of MLLMs, we believe that leveraging their video understanding capabilities can assist us in labeling. Specifically, we use `gpt-4o` based on video input to attempt to describe all the game glitches that appear in the video.

Input Preparation Based on practical usage, we find that when the number of input frames exceeds a certain threshold, the model's video understanding ability fluctuates. We believe this is due to the excessive number of input images preventing the model from focusing on specific details. Thus, we limit the number of input frames to fewer than 20 and add frame indices as the textual hint to the MLLM. Specifically, we split the video into segments shorter than ten seconds and obtain sampled frames at a rate of 2 frames per second (FPS).

Additional Hints Identifying glitches in a short gameplay video can be quite challenging for people who haven't played it before. Utilizing discussions about the glitched videos on Reddit (including titles and conversations) can be very helpful in identifying and describing glitches. We select 100 instances of data along with their corresponding generated pseudo descriptions and manually assess their accuracy. The results from Table 3.2 indicate that allowing the MLLM to access these relevant discussions is very beneficial for generating glitch descriptions.

Method	**Accuracy (%)**
Baseline	50.51
+ Reddit Discussion	**64.29** (+13.78)

Table 3.2: Accuracy improvement by adding Reddit discussion

3.3 Human Validation & Temporal Annotation of Glitches

We have developed a multi-user annotation interface 3.2 that allows multiple annotators to annotate different videos simultaneously. For many videos, annotators can directly copy the glitch descriptions generated by the MLLM as the ground truth. However, other videos may require some modifications or additions in case the MLLM cannot accurately identify the glitches in the video or correctly describe them, such as missing important elements or describing incorrect characters.

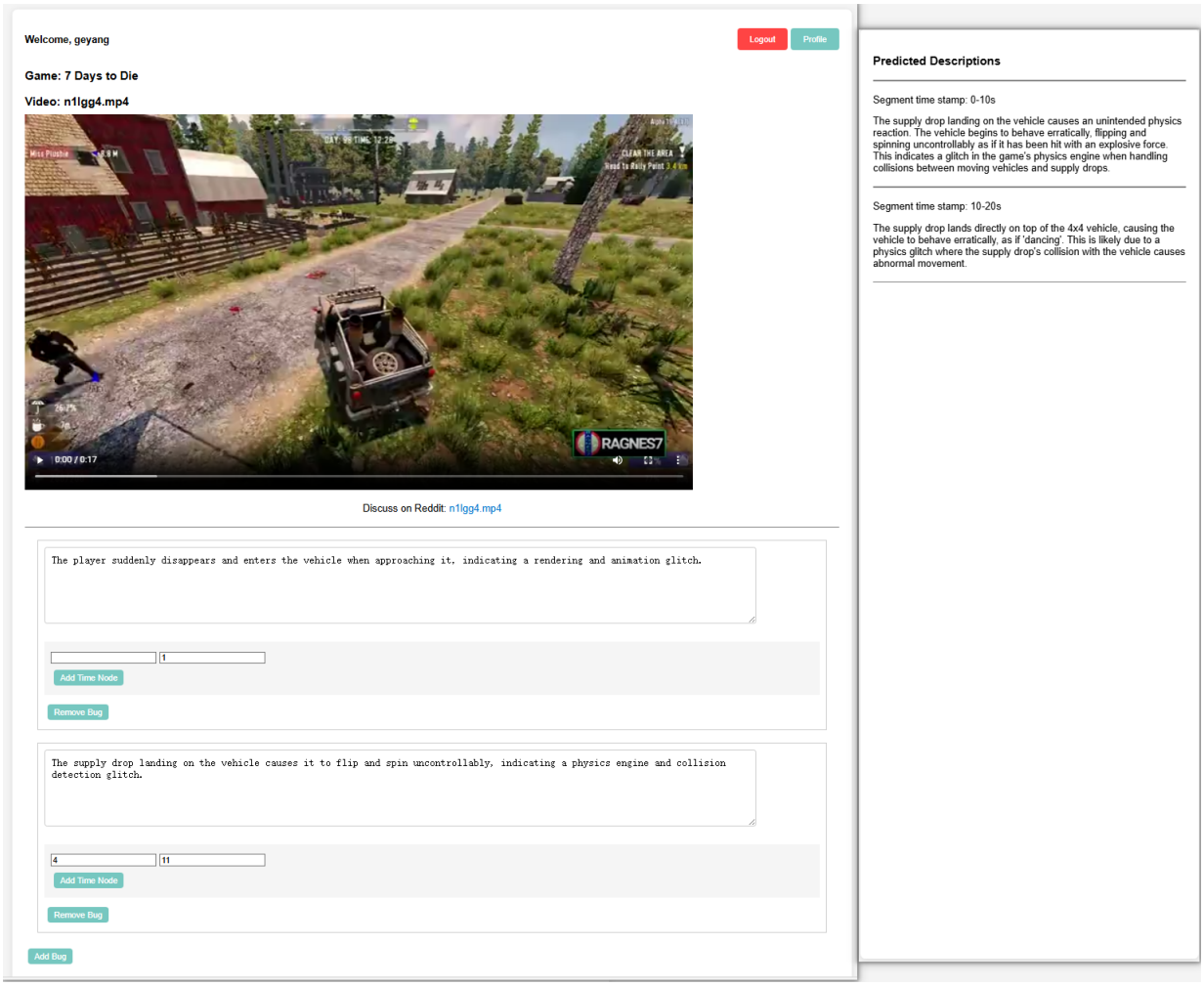


Figure 3.2: Human Annotation Interface

While watching the video, the start and end timestamps of each glitch description will be marked at the corresponding position in the interface by the annotator.

Chapter 4

Methodology

This chapter details our multi-stage methodology for automatically detecting, verifying, and documenting glitches in gameplay videos using a Multimodal Large Language Model (MLLM). The framework consists of four primary parts, as shown in Figure 4.1. First, the Video Preprocessing stage (Sec. 4.1) transforms raw video into a computationally manageable format by uniformly sampling frames at a fixed rate and segmenting them into discrete, non-overlapping windows. Second, the Window-based Glitch Hypothesis Generation stage (Sec. 4.2) uses an MLLM to analyze each window independently and then group them into glitch records, generating an initial, sensitive list of potential glitches. Finally, the framework traces the full duration of each confirmed glitch through Temporal Propagation (Sec. 4.3), using verified instances as "anchors" to search adjacent windows, and concludes with Canonical Description Generation (Sec. 4.4), where the MLLM synthesizes all related information into a single, cohesive summary for each unique glitch event. The final output is a structured report detailing each glitch with a precise timeline and a definitive description, ready for evaluation.

4.1 Video Preprocessing

The primary input to our system is the raw video of the gameplay sessions. However, an MLLM does not process a video stream continuously; rather, it analyzes a sequence of

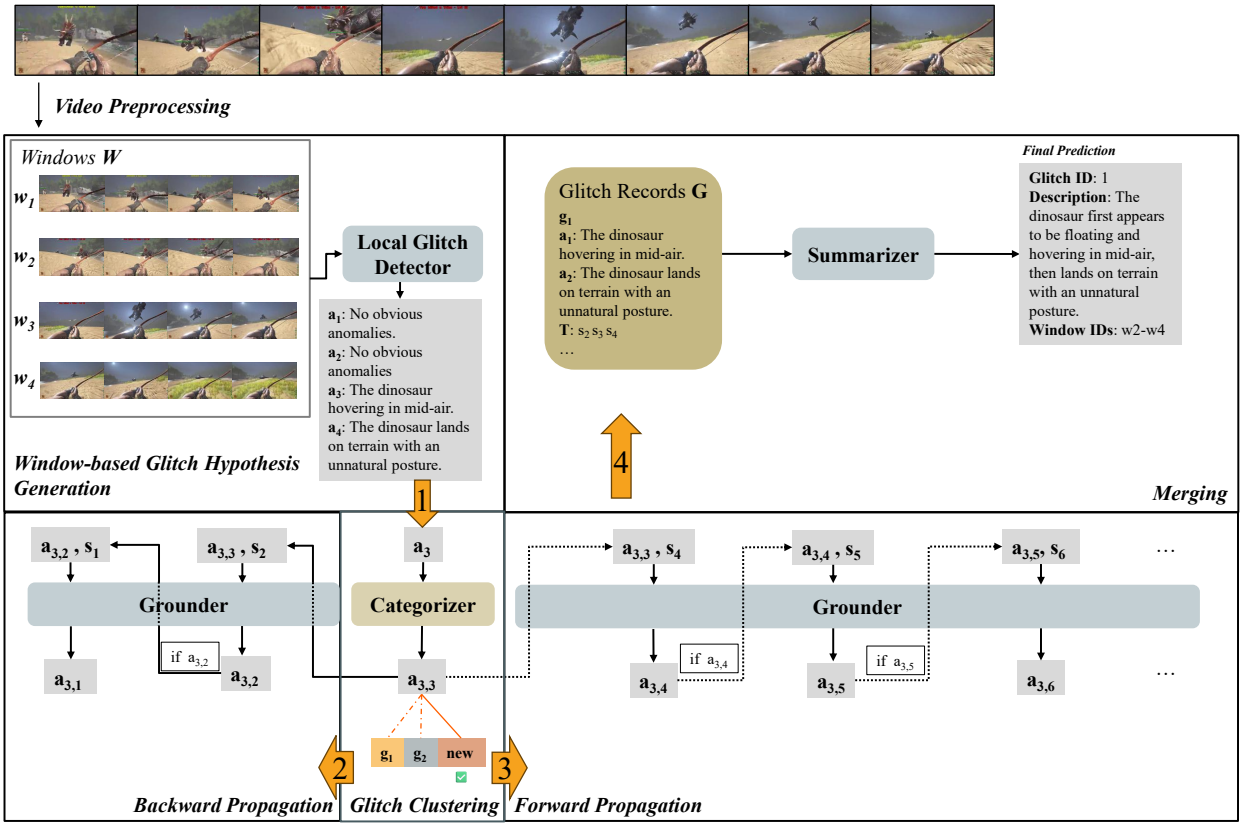


Figure 4.1: Overview figure of our GlitchAgent.

discrete image frames. The objective of the video preprocessing stage is, thus, to convert this continuous video footage into a structured sequence of discrete frames that are representative of the gameplay and computationally tractable for the MLLM.

Following established video analysis practices, we adopt a uniform sampling strategy. It consists of choosing individual frames from the video at a fixed-and-constant time interval. We have given more importance to the uniform sampling method due to its simplicity and efficiency, and that it gives an undistorted chronological view of the state of the game. Sampling-wise, if we keep a constant interval over time, an important event, say a glitch, cannot entirely be missed out on by methods emphasizing focus only during moments of high visual accumulation.

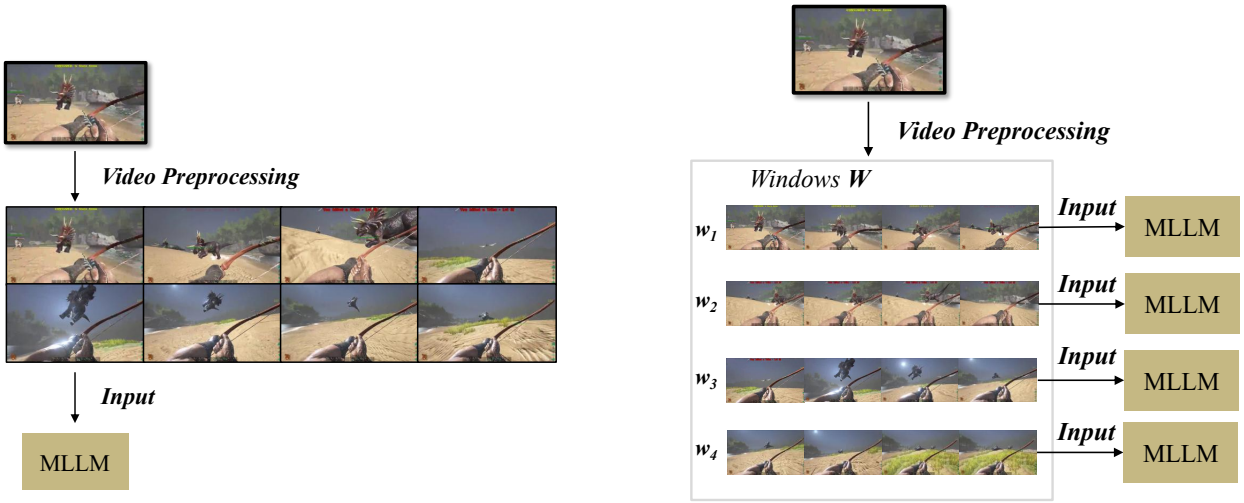
In our framework, we define a sampling rate, f_s. For this study, f_s is set to 1 frame per second (FPS). This rate is chosen to provide a consistent and sufficiently detailed overview of gameplay events while mitigating the high data redundancy present in high-frame-rate video (e.g., 60 FPS), where consecutive frames are often visually almost identical, while the objective of glitch detection is to search for anomalous changes.

Our selection of 1 FPS is hypothesized to strike an effective balance between comprehensive temporal coverage and respecting the MLLM's processing capabilities. The precise impact of this sampling rate on the final glitch detection performance is a key experimental variable. Specific experimental comparisons and a quantitative justification for this choice will be presented in Sec. 5.2.2. This will validate our approach by comparing the efficacy of different sampling rates.

Once the sequence of frames is extracted at sampling rate f_s, it is divided into non-overlapping windows so that one such window can be fed into the MLLM as a discrete input. Each window contains a fixed number of consecutively sampled frames, which we refer to as the window size, w_s. The choice of w_s is an important hyperparameter because it determines the length of the context the model can consider when queried in a single inference step.

Previous sampling methods often use randomly uniform sampling of 8 or 16 frames, which can result in a great loss of temporal information amongst consecutive frames. Furthermore, these methods would feed all frames into the MLLM at once, thus creating a pressure on the context length limitation and the comprehension ability of MLLM. However, in our method, the frames are sampled at a fixed FPS, contributing to the preservation of temporal information, and granting the LMM detailed analysis within each local temporal window. Sampling methods are distinguished in Figure 4.2.

A larger w_s provides the model with more information to identify glitches that unfold over



(a) Previous sampling methods.

(b) Sampling method in our framework.

Figure 4.2: Comparison of different sampling methods.

several seconds, while a smaller w_s allows for more granular, localized analysis but may miss longer-term contextual cues. A quantitative analysis of the model's performance under different values of w_s will be presented in Sec. 5.2.2 to empirically determine the optimal window configuration for our task.

4.2 Window-based Glitch Hypothesis Generation

Following the video preprocessing stage, the raw video has been transformed into a temporally ordered sequence of non-overlapping windows, where each window contains w_s frames. This section details the core process of generating initial glitch candidates, which we refer to as glitch hypotheses. This is achieved by feeding each window into a component we term the local glitch detector.

As the name suggests, the local glitch detector deals with every window in isolation. Given each window, which in and of itself is a sequence of w_s frames, the MLLM is prompted

to carry out a local analysis. The agent should scrutinize the short segment in question for visual and temporal information, determine if any anomalous behaviors, visual artifacts, or unexpected game-state transitions typical of the glitch occur there, and make its choice. This choice for each window is binary (glitch or non-glitch), with the model providing a short rationale in free text for the suspicion of a glitch and a detailed description of the glitch if the suspicion is confirmed.

The reasoning for this localized, window-by-window method is a two-way objective for maximizing both the sensitivity and robustness:

Enhanced Focus on Local Anomalies The rationale is that by restricting the model's analytical scope to a short and isolated segment, subtle and brief anomalies can be detected more effectively. During global analysis of a long video sequence, a few anomalous frames representing a momentary glitch (e.g., texture pop-in, a character model clipping through a wall for a frame) can be easily diluted or overshadowed by the large amount of footage depicting normal gameplay. Within a local window of more immediate influence, such an event gains weight as an intolerable entity influencing the greater portion of its input; hence, it is much more salient and difficult for the model to ignore. This way, localized glitches cannot simply become averaged out with the "big-picture view."

Increased Detection Probability for Persistent Glitches If a glitch persists for a duration longer than a single window (i.e., longer than w_s/f_s seconds), it will be present across multiple, consecutive windows. By processing each window independently, our system gains multiple opportunities to detect the same underlying issue. This temporal redundancy significantly increases the overall probability of detection. A single false negative on one window is not a catastrophic failure, as the detector has subsequent chances to identify

the same persistent anomaly in the following windows. In addition, the same glitch may intermittently appear in the video, and our method enhances the accuracy of temporal localization by executing detection in a more refined manner.

The immediate output after acting with a local glitch detector on each window in the video is a list of windows possibly containing glitches. We call this curated list the local glitch hypotheses. Each hypothesis denotes a specific time slot in the source video and stands as a very promising candidate for the presence of a real glitch. In this initial pass, the whole gameplay footage is effectively screened, and thus, the problem changes from scouring exhaustively through thousands of frames to looking into a small set of interesting events: suspicious windows.

Upon the creation of the hypothesis, this phase also conducts glitch categorization and groupings as a second major step. A raw list of verified glitches may be accurate, yet it is far from being made useful for any type of analysis. Therefore, the moment a glitch is verified, the MLLM is set to perform a classification whereby it compares the verified event against a known glitch record induced from earlier window(s). This means the task of the model is to determine if it can semantically group the current glitch instance with a record of known glitches.

This finest set of tools is used for two objectives: First, to label the glitches uniformly to allow further aggregation of similar or relevant cases. Second, for preparing the data for the final tracking phase, during which the temporal locations and types of unique glitches will be tallied across the entire gameplay session.

The output of the Window-based Glitch Hypothesis Generation stage is a structured list of grouped glitch incidents. Each entry in this list contains the timestamp, the window ID, a precise textual description of the glitch event, and its assigned a unique glitch record ID

from our glitch records memory.

4.3 Temporal Propagation from Glitch Anchors

Our system, as elaborated in the hypothesis generation stage in Section 4.2, has produced a high-confidence and yet potentially incomplete list of glitch occurrences. The window-based hypothesis generation is sensitive but it does analysis in isolation for each window. Such a discrete analysis may likely introduce false negatives during the initial or trailing period of a glitch event when the subliminal signs cannot compel an initial unguided detection. Hence, these grouped glitches serve as reliable **"anchor points"** but may not span the complete time duration of an anomaly. To overcome the aforementioned drawback, this stage, called Temporal Propagation, systematically expands from those grouped anchors to track the entire duration of every glitch.

The core principle of this stage is to leverage the confirmed glitch descriptions as a powerful form of cognitive scaffolding for the MLLM. Instead of asking the model to perform another open-ended search for *any* glitch in adjacent windows, we pose a much more constrained and targeted question: "Does the specific glitch, described as '[verified glitch description]', also occur in this window?" This transforms the task from discovery to targeted propagation, significantly increasing the probability of detecting fainter manifestations of the same event that were previously overlooked. This propagation process operates bidirectionally from each anchor, comprising two phases: backward propagation to identify the event's onset and forward propagation to determine its conclusion.

Backward Propagation from the Earliest Anchor Starting with a unique glitch record, backward propagation is initiated from whatever window is in that record and

marked as the earliest confirmed window, i.e., the initial anchor. Let us denote this early window ID as w_i. What follows is an iterative examination of the windows going backward $(w_{i-1}, w_{i-2}, \dots)$. For each window under scrutiny, the Grounder is queried about the presence of the glitch described specifically in that record. Backward propagation for this particular glitch record continues until either of the two confinement criteria arises:

1. **Negative Detection:** The Grounder has announced that the specific glitch is no longer present in the window under examination, which indicates we have found the temporal onset of a glitch event.

2. **Record Collision:** The propagation encounters a window already confirmed in the same glitch record. This condition is important for efficiency so that we do not repeat analysis of the same glitch occurrence.

Any newly positively identified windows are added to that glitch record, thus extending the known duration of the glitch backward in time.

Forward Propagation from the Latest Anchor Forward propagation, much like backward propagation, strives to zero in on the particular instant when the glitch terminates. It starts immediately after the backward propagation and proceeds from the anchor window, w_i, through subsequent windows $(w_{i+1}, w_{i+2}, \dots)$. At every step, the Grounder is issued the targeted verification task. Similar to the backward operation, the forward propagation terminates at one of two instances: upon a negative detection that affirms the glitch has indeed ended or when it collides with a pre-existing window within the same glitch record.

By extending this preparatory bidirectional propagation from every grouped anchor, we achieve a complete temporal mapping for each distinct glitch. This procedure goes beyond a mere setting of independent detections, in favor of assembling a contiguous timeline that

binds adjacent windows that share the same anomalous behavior. The final outcome of this stage is a consolidated set of all glitch records, temporally complete. Each record shall contain an ID for the unique glitch, a set of temporal descriptions, and a complete list of all window IDs in which this glitch has manifested, giving an almost exact timeline concerning when it appeared from the onset towards the offset.

4.4 Canonical Description Generation and Final Merging

Following the temporal propagation stage (Sec. 4.3), our system has successfully mapped the full temporal extent of each unique glitch. The output is a set of glitch records, where each record contains a unique ID and a complete, contiguous list of all window IDs in which the glitch manifests. However, a single glitch record may be associated with multiple textual descriptions—one for each of the initially detected "anchor" windows. These individual descriptions, while confirmed, can be fragmented, redundant, or may only capture a specific aspect of an evolving glitch. The objective of this final stage is to synthesize these multiple, detected descriptions into a single, comprehensive, and definitive canonical description for each glitch record.

The main task here is that of contextual summarization-a very interesting task for which LLMs are well equipped. Simply sticking existing descriptions together would give a rambling and disjointed story. Instead, we use the Summarizer as a reasoner on the textual descriptions to produce a higher-level, holistic summary.

Each unique glitch record is yielded from one record in the following way. In step one, we gather all individual, confirmed textual descriptions associated with that record. All

these descriptions are given to the Summarizer. Finally, the Summarizer is now being asked to perform synthesis; 'Review the provided video game glitch descriptions in time order. Generate a single, concise, and accurate canonical description that summarizes the entire glitch event from its beginning to its end. Remove redundancy and resolve any minor inconsistencies between the initial descriptions to create the most representative summary.'

The model can observe how the anomaly begins, whether it evolves or remains static, and how it ends. From this, the final canonical description can be both semantically rich and accurate, representing what constitutes the essence of a glitch. For instance, should a character model start clipping in a wall, and sometime later it starts vibrating uncontrollably, the final description may well convey the entire sequence of events.

Thus, the output of this final merging stage is a fully refined and consolidated set of glitch reports. Each report consists of:

1. A unique Glitch ID for traceability.
2. A complete list of all window IDs, states the exact start and end time of the glitch.
3. A single LLM-produced canonical description, which is the definitive and human-readable summarization of the problem.

From the end, this structured output stands for the end product of our detection pipeline, turning raw video footage into a list of well-documented and temporally-localized glitches that can be reviewed by QA engineers.

Chapter 5

Experiments

This chapter is mainly divided into two sections: Experimental Setup (Sec. 5.1) and Experimental Results (Sec. 5.2).

5.1 Experimental Setup

5.1.1 Evaluation Metrics

Evaluating the performance of our video game glitch detection system requires a metric that assesses two distinct but equally important facets: the temporal accuracy of the detection (i.e., when the glitch occurred) and the descriptive accuracy of the generated report (i.e., what the glitch was).

A parallel can be drawn to the task of Dense Video Captioning, where the goal is to generate a sequence of descriptive text events, each with a corresponding non-overlapping timestamp. A prominent evaluation framework in this domain is the Story-Oriented Dense video cAptioning evaluation framework (SODA) [7]. SODA is designed to holistically measure performance by considering both the quality of generated captions and the localization accuracy of the events. It achieves this by first finding correspondences between generated and ground-truth events and then calculating a final score where the text similarity metric (typically METEOR) is weighted by the temporal Intersection over Union (IoU) – a common metric

for temporal grounding task – of each matched pair. However, a direct application of the SODA framework to our glitch detection task presents two significant challenges that limit its effectiveness:

1. **Sub-optimal Matching Strategy**: SODA typically employs a temporally greedy or locally optimal matching algorithm. It matches a generated caption to the ground-truth caption that has the highest temporal IoU, provided it meets a certain threshold. This approach is ill-suited for glitch detection, where multiple, distinct glitches might occur within the same or overlapping time windows. For instance, a character model T-posing (a visual glitch) could happen concurrently with the character clipping through the floor (a physics glitch). A purely temporal matching could incorrectly pair the ground-truth description of "clipping" with the system's generated description of "T-posing," leading to an inaccurate evaluation of the system's descriptive capabilities.

2. **Inadequate Text Similarity Metric**: The SODA framework commonly relies on the METEOR score to evaluate caption quality. METEOR is based on n-gram matching, stemming, and synonymy. While effective for general-purpose sentence similarity, it fails to capture the semantic nuance required for technical glitch descriptions. For example, the descriptions "The character's arm is clipping through the wall" and "Player model mesh is interpenetrating with the level geometry" are semantically identical in a QA context, but their low lexical overlap would result in a poor METEOR score. This penalizes valid, descriptive reports that do not precisely match the ground-truth verbiage.

Having overcome those limitations with SODA, we propose an evaluation framework, LLM-Judged Glitch Detection Score (GDS), that refactors SODA's core principles but replaces certain elements that do not work well in our context with others that are better suited. Our

metric is therefore designed to better align with human judgment. The evaluation is carried out through the following four steps:

Stage 1: LLM-as-Judge Semantic Scoring Instead of relying on lexical matching, we employ a powerful Large Language Model (LLM) as an impartial judge to score the semantic similarity between each generated glitch description and each ground-truth description. For every possible pair of a generated report (g) and a ground-truth report (gt), the LLM is prompted to provide a similarity score on a scale of 0 to 5, where 0 indicates no relation and 5 indicates semantic equivalence. This captures the true meaning of the description, overcoming the limitations of METEOR. We manually evaluate 100 random samples, as illustrated in table 5.1. The findings robustly indicate that the Pearson correlation coefficient between GDS and human scores is significantly superior to that of the METEOR score.

Metric	**Pearson Correlation**
METEOR score	0.396
LLM score	**0.723**

Table 5.1: Correlation comparison with human score

Stage 2: IoU Score Computation While the semantic score measures the *what* of the glitch, this stage is all about the *where*. Checking the precision of a location means that we use the Intersection over Union (IoU) measure between the predicted time ranges (T_p) and the ground-truth time ranges (T_{gt}). This IoU score, a very frequently used metric in temporal grounding, is meant to express the extent to which two time ranges overlap. It is the ratio of the area of time range intersection to the area of the union:

$$IoU = \frac{\text{Area}(T_p \cap T_{gt})}{\text{Area}(T_p \cup T_{gt})}$$

This measure would be between 0 and 1, with the value 1 used to describe a perfect match; on the other hand, if the value is zero, it means there is no overlap. A very high IoU score indicates that the glitch has been very well localized by the model, and hence it serves as a temporal counterpart to the Stage 1 semantic score.

However, the IoU score is not meaningful in isolation; a high IoU might be assigned to a prediction whose description is semantically unrelated to the ground-truth event. To address this, we combine semantic relevance with temporal precision by calculating an **LLM-weighted IoU score**. This score is computed by multiplying the LLM similarity score (S_{LLM}) from Stage 1 with the IoU score, ensuring that high temporal accuracy is only rewarded when the semantic description is also correct. Consequently, we produce two intermediate metrics: the semantic similarity LLM score itself, and the LLM score-weighted IoU for a holistic measure of temporal accuracy.

Stage 3: Global Optimal Matching via the Hungarian Algorithm With a complete similarity matrix containing the LLM scores for all possible (g, gt) pairs, we use the Hungarian Algorithm to find the globally optimal assignment. This algorithm identifies the set of one-to-one matches that maximizes the total similarity score across all pairs, ensuring that each generated glitch is matched with its most semantically relevant ground-truth counterpart, rather than the one that is closest in time. We perform the global optimal matching operation on both the obtained semantic LLM score.

Stage 4: Final Metrics Computation After establishing the optimal matched pairs, we follow the SODA methodology for the final calculation. We derive F-measure scores from both LLM score and LLM score-weighted IoU score to penalize redundant/irrelevant descriptions. Thus, we use Precision, Recall and F1 to measure the quality of generated

glitch descriptions, we use mean matched IoU to measure the average grounding accuracy of predicted time ranges, and we use an F1×IoU score to measure the overall performance.

By replacing local, temporal matching with global, semantic matching and substituting METEOR with a more robust LLM-as-judge score, our proposed GDS metric provides a more accurate and meaningful evaluation of our system's performance. As we will demonstrate in our results, this metric shows a significantly higher correlation with human preference scores compared to a direct application of the SODA framework, confirming its superiority for the nuanced task of video game glitch detection.

5.1.2 Implementation Details

Our framework is implemented using the baseline models such as Qwen2.5-VL-3B-Instruct in a zero-shot setting, without any additional training. For video preprocessing, input frames are resized to a fixed resolution of 336×336 pixels. During inference, we set the decoding temperature to 0.5. These settings were applied uniformly across all models, including the baselines, to ensure a fair comparison. To guarantee valid and parsable output, we utilize structured prompting to compel the model to generate glitch descriptions paired with their corresponding timestamps.

Experiments were conducted using A100s (80GB) or H200s (141GB) GPUs for open-source models, and API calling for proprietary models.

5.1.3 Baseline Models

We evaluate both open-source and proprietary models.

1. **Proprietary Models**

(a) gpt-4o-mini [14], a small model with superior textual intelligence and multimodal reasoning developed by OpenAI.

(b) gemini-2.0-flash [8], it surpasses the response speed of the previous generation model by two times and supports both multimodal input and output.

(c) claude-3.5-haiku [3], is the next generation of our fastest model. For a similar speed to Claude 3 Haiku, Claude 3.5 Haiku improves across every skill set and surpasses even Claude 3 Opus, the largest model in our previous generation, on many intelligence benchmarks.

(d) nova-lite-v1 [1], is a low-cost multimodal model that is lightning fast for processing images, video, documents and text.

2. **Open-source Models**

(a) Qwen2.5-VL 3B/7B Instruct [4], the latest flagship model of Qwen vision-language series, which demonstrates significant advancements in both foundational capabilities and innovative functionalities, especially in visual recognition and long-video comprehension.

(b) Intern2.5-VL 4B/8B [6], an advanced series of MLLMs that builds upon the foundational framework of InternVL 2.0, simultaneously introduces substantial improvements in both training and testing methodologies, as well as an elevated standard of data quality.

(c) Kimi-VL-A3B-Instruct [19], an efficient open-source Mixture-of-Experts (MoE) vision-language model (VLM) that offers advanced multimodal reasoning, long-context understanding, and strong agent capabilities.

(d) UI-TARS-1.5-7B [15], an open-source multimodal agent built upon a powerful vision-language model. It is capable of effectively performing diverse tasks within

virtual worlds.

(e) LLaVA-OneVision-7B [9] is the first single model that can simultaneously push the performance boundaries of open LMMs in three important computer vision scenarios: single-image, multi-image, and video scenarios.

(f) InternVideo2_5-8B [21] is a new version of InternVideo series model with a focus on enhancing the original MLLMs' ability to perceive fine-grained details and capture long-form temporal structure in videos.

5.2 Results

This section mainly presents the evaluation performance of various baselines and our framework on our benchmark. In Sec. 5.2.1, we present the performance of all baselines and compare them with our framework. In Sec. 5.2.2, we analyze the impact of different modules and hyperparameters.

5.2.1 Main Results

Baseline Performance Analysis The evaluation results of baseline models on our benchmark are demonstrated in Table 5.2. A recurring issue is evident in the performance results: notably low precision scores. This arises from the generation of numerous low-scoring or even zero-scoring glitch descriptions, often accompanied by substantial hallucinations regarding temporal understanding. For instance, the model may report abnormal phenomenon about vehicle's explosion in the video where no vehicle-related issues are present, indicating a misinterpretation of the content, as shown in Figure 5.1.

Model	Description Generation			Temporal Grounding	
	Precision (%)	**Recall (%)**	**F1 (%)**	**mIoU**	**F1 × IoU (%)**
Proprietary Models					
gemini-2.0-flash	18.36	25.35	21.29	0.44	10.55
gpt-4o-mini	12.33	32.36	17.86	0.35	6.20
claude-3.5-haiku	21.66	32.54	26.01	0.47	12.91
nova-lite-v1	6.66	23.77	10.41	0.28	2.98
Open-source Models					
Qwen2.5-VL-3B-Instruct	11.08	26.53	15.63	0.31	4.81
Qwen2.5-VL-7B-Instruct	10.41	14.74	12.20	0.30	4.11
InternVL2_5-4B	8.62	22.93	12.53	0.18	1.92
InternVL2_5-8B	11.90	23.36	15.77	0.25	4.06
Kimi-VL-A3B-Instruct	3.90	16.29	6.29	0.24	1.67
UI-TARS-1.5-7B	19.08	24.93	21.62	0.40	9.11
LLaVA-OneVision-7B	5.9	19.5	9.06	0.26	2.18
InternVideo2_5-8B	17.8	20.13	18.9	0.29	5.85

Table 5.2: Baseline Models Comparison

Figure 5.1: **Ground Truth**: Dead body lying on the road slides rapidly. **Prediction**: An unexpected bullet hits the player's vehicle, causing it to explode.

Model	Description Generation			Temporal Grounding	
	Precision (%)	**Recall (%)**	**F1 (%)**	**mIoU**	**F1 × IoU (%)**
Proprietary Models					
gpt-4o-mini	12.33	32.36	17.86	0.35	6.20
+**GlitchAgent**	44.69	43.91	44.30	0.56	25.78
Open-source Models					
Qwen2.5-VL-3B-Instruct	11.08	26.53	15.63	0.31	4.81
+**GlitchAgent**	33.21	26.72	34.88	0.56	19.83
Qwen2.5-VL-7B-Instruct	10.41	14.74	12.20	0.30	4.11
+**GlitchAgent**	28.04	19.10	22.73	0.53	12.48
InternVL2_5-4B	8.62	22.93	12.53	0.18	1.92
+**GlitchAgent**	27.03	22.49	24.55	0.54	13.89
InternVL2_5-8B	11.90	23.36	15.77	0.25	4.06
+**GlitchAgent**	32.57	35.06	33.30	0.54	18.68
UI-TARS-1.5-7B	19.08	24.93	21.62	0.40	9.11
+**GlitchAgent**	24.99	27.21	26.05	0.54	14.28
LLaVA-OneVision-7B	5.9	19.5	9.06	0.26	2.18
+**GlitchAgent**	28.24	20.87	24.01	0.55	13.63

Table 5.3: Baseline models performance compared with adding our framework.

Table 5.4: Comparison of baseline, GlitchAgent, and ground truth results.

Video	Baseline	GlitchAgent	Ground Truth
	Prediction 1: Entity appears to **float** above the ground unexpectedly. (5–10s) Prediction 2: Entity rapidly changes direction without animation. (10–14s)	Prediction 1: ..., the creature (likely a **Triceratops**) exhibits an abnormal behavior where it appears to be **flying or levitating** off the ground, ...suggests a glitch in its animation or **physics** engine, as it should be running along the ground instead of being airborne. (0s-14s)	After the player kills the Triceratops ...it starts to levitate unrealistically, ascending into the air and moving horizontally ...it defies the expected physics of the game world ...(2s-14s)

Our Method Table 5.3 shows a comprehensive comparison of baseline performance with and without the GlitchAgent framework in various private and public vision-language models. It can be observed from the experimental result that the GlitchAgent framework significantly benefits description generation and temporal grounding tasks, with the F1 score improving by 13.17 on average, mIoU by 0.25 on average, and F1×IoU by 12.31 on average. In particular, proprietary models such as gpt-4o-mini show a notable improvement in the F1 score for description generation, going from 17.86% to 44.30%, and corresponding temporal grounding performance with an mIoU improvement from 0.35 to 0.56 and F1×IoU improvement from 6.20% to 25.78%. Similar trends can be seen among open-source models; in particular, the F1 score of Qwen2.5-VL-3B-Instruct increased from 15.63% to 34.88%, and the F1×IoU even surged fourfold, from 4.81% to 19.83%. Moreover, all evaluated models observe the GlitchAgent framework as a blessing, with almost all other metrics (precision, recall, F1, mIoU) consistently increasing. These results shed light on the success and generalizability of our proposed method and provide insight into its potential to improve description generation correctness and temporal localization across varied multimodal models.

Table 5.4 shows a qualitative comparison demonstrating the greater analytical capacities

of the GlitchAgent with regards to the baseline. The baseline model announces two separate and temporally disjointed events: an entity "floating" from 5 to 10 seconds and then rapidly changing direction between 10 and 14 seconds. Though on the surface, there may be some truth in this statement, it is rather fragmented and lacks depth because it fails to comprehend these as two perspectives of a single continuous underlying issue. GlitchAgent is capable of a more holistic and insightful analysis, correctly merging the two observations into one event spanning from 0 to 14 seconds. Furthering the description with crucial contextual data, GlitchAgent identifies the entity as a "Triceratops" and attributes the anomalous behavior of "flying or levitating" to an equally plausible cause, i.e., a "glitch in its animation or physics engine." In essence, this elaborate interpretation very closely agrees with the human-annotated ground truth, which similarly describes this event as a creature that "levitate[s] unrealistically" or "defies the expected physics." In essence, this example illustrates GlitchAgent's ability to move beyond simple event detection capabilities into a more advanced understanding of in-game anomalies that are contextually aware and mirror human-like causal reasoning.

5.2.2 Ablation Studies

Impact of Sampling Rate

To check for temporal resolution influence, we conduct an ablation by changing the frame sampling rate, as shown in Table 5.5. Four rates such as 2.0, 1.0, 0.5, and 0.25 FPS are researched to shed light on how much each model respond to temporal granularity with greater resolution.

The baseline Qwen2.5-VL-3B-Instruct model does not benefit from higher frame rates. Its F1 score for description generation fluctuates without a clear upward trend, and does not

Model	FPS	Description Generation			Temporal Grounding	
		Precision (%)	Recall (%)	F1 (%)	mIoU	F1 × IoU (%)
Qwen2.5-VL-3B-Instruct	2.0	10.11	24.85	14.37	**0.35**	**5.06**
	1.0	11.08	26.53	**15.63**	0.31	4.81
	0.5	10.81	24.65	15.03	0.29	4.52
	0.25	11.38	23.30	15.29	0.28	4.16
Qwen2.5-VL-3B-Instruct + **GlitchAgent**	2.0	31.75	39.76	**35.31**	0.55	19.66
	1.0	33.21	26.72	34.88	0.56	**19.83**
	0.5	33.17	33.32	33.25	**0.53**	18.00

Table 5.5: Performance comparison of Qwen2.5-VL-3B-Instruct based on different sampling rates.

improve at the highest sampling rate of 2.0 FPS. The model's localization accuracy, measured by mIoU, steadily decreases as the sampling rate drops—from 0.35 at 2.0 FPS to 0.28 at 0.25 FPS—It aligns with intuition, as the lower the FPS, the greater the time interval between adjacent frames, resulting in more temporal information lost.

In contrast, our proposed method (Qwen2.5-VL-3B-Instruct + GlitchAgent) exhibits consistently strong performance as the frame rate increases, which proves that our method enhances the model's utilization of more detailed information. The F1 scorereach its highest values at 2.0 FPS while the mIoU and joint F1 × IoU metric reach their highest values at 1.0 FPS, indicating that the model can fully capitalize on denser temporal input to enhance both descriptive quality and temporal localization. Notably, at reduced sampling rates, the expected trade-off between Precision and Recall emerges: Precision declines slightly while Recall increases; however, the model maintains robust localization accuracy, with mIoU holding steady at 0.53–0.56.

Hence, the results demonstrate that our approach benefits from increasing the frame density to translate the fact into a very good performance, something the baseline model can never do. The results show that 2.0 FPS can provide the best tradeoff between event detection and temporal localization accuracy with our methodology.

Impact of Window Size

Model	Window Size	Description Generation			Temporal Grounding	
		Precision (%)	Recall (%)	F1 (%)	mIoU	F1 × IoU (%)
Qwen2.5-VL-3B-Instruct+**GlitchAgent**	4	27.65	38.45	32.16	0.55	18.42
	8	31.41	39.19	34.87	0.55	19.41
	16	33.21	26.72	**34.88**	**0.56**	**19.83**

Table 5.6: Performance of Qwen2.5-VL-3B-Instruct+GlitchAgent with varying window sizes.

Our ablation tests investigate how the temporal context length influences the model's behavior based on the window size of the inputs. Window sizes of 4, 8, and 16 frames are tested with the performances reported in Table 5.6.

Table 5.6 shows a definite trend with respect to the changes in the context length. When the window increases from 4 to 16, a corresponding improvement in Precision is evident-from 27.65% up to a maximum of 33.21%. This indicates that a higher temporal context facilitates its model to come up with more confident and accurate event descriptions. On the other hand, Recall is highest at windows of size 8 (39.19%) and falls at 26.72% for size 16, indicating a possible trade-off where longer contexts might fail to spot shorter events in favor of more prominent ones.

The model is therefore most effectively summarizing all performances for a window size of 16. This window size produces the highest description generation F1 score (34.88%), the best temporal localization mIoU (0.56), and the highest combined F1 × IoU score (19.83%). Although the F1 score at window size 16 is almost the same as that at window size 8 (34.87%), given that it performs much better on temporal grounding metrics, it is the clear choice for the best option. For this reason, future experiments with our method will use window size 16 as the default, given that it provides the best trade-off of description generation and temporal grounding accuracy.

Importance of Temporal Propagation

Model	mIoU
gpt-4o-mini+GlitchAgent	0.56
w/o propagation	0.35
Qwen2.5-VL-3B-Instruct+GlitchAgent	0.56
w/o propagation	0.31
Qwen2.5-VL-7B-Instruct+GlitchAgent	0.53
w/o propagation	0.30
InternVL2_5-4B+GlitchAgent	0.54
w/o propagation	0.18
InternVL2_5-8B+GlitchAgent	0.54
w/o propagation	0.25
UI-TARS-1.5-7B+GlitchAgent	0.54
w/o propagation	0.40
LLaVA-OneVision-7B+GlitchAgent	0.55
w/o propagation	0.26

Table 5.7: Comparison of mIoU with and without temporal propagation across different models.

To demonstrate the critical role of our proposed temporal propagation mechanism, we conducted an ablation study across seven different backbone models. As presented in the accompanying table, we compare the temporal grounding performance of our full GlitchAgent against a variant where the propagation mechanism is disabled (w/o propagation).

The results unequivocally highlight that the temporal propagation mechanism is indispensable for accurate temporal localization. Across all tested models, its removal leads to a substantial degradation in the mIoU. The most dramatic drop is observed with the InternVL2_5-4B backbone, where the mIoU plummets from 0.54 to 0.18—a relative reduction of 66.7%. Similar significant decreases are seen with other models; for instance, the mIoU for LLaVA-OneVision-7B is more than halved, falling from 0.55 to 0.26 (a 52.7% relative decrease), and for Qwen2.5-VL-3B-Instruct, it drops from 0.56 to 0.31.

The massive degradation in performance implies that these models analyze the video seg-

Table 5.8: Comparison of GlitchAgent, w/o propagation results.

Video	GlitchAgent	w/o Propagation	Ground Truth
	Prediction 1: The dragon-like creature appears to be floating or hovering in the sky after being killed without any apparent means of support. This defies the expected physics of the game. (0s-14.5s) IoU=0.9	Prediction 1: The dragon-like creature appears to be floating or hovering in the sky after being killed without any apparent means of support. This defies the expected physics of the game. (2.0-3.5s) IoU=0.11	The Argentavis remains hovering in the air after being killed, instead of falling to the ground as expected due to gravity. After that, it fell to the ground stuttering. This suggests a glitch in the game's physics engine where the creature is not responding to gravitational forces properly. (1s-15s)

ments in isolation and, hence, miss the very temporal context needed to follow the full duration of an event. Our method ensures that consecutive windows of analysis are bridged so that the state may be maintained, allowing the model to align with the initial and end times of events that stretch over two or more segments. Provided that these improvements are consistent and massive across variants of model architectures, temporal propagation presents itself as a very fundamental and resilient ingredient underpinning our approach, required for high-performance temporal groundings.

To shed light on the irreplaceable involvement of the propagation module in GlitchAgent, a dedicated ablation study was conducted, with the results compared in Table 5.8. It analyzed one specific in-game glitch, where the glitch makes a creature defy physics by adequately hovering in the air post death. Our full GlitchAgent model offers a very efficient performance that correctly places the start of the glitch in a continuous interval from 0 to 14.5 seconds, with an IoU of 0.90 with the ground truth. Conversely, without the propagation module, the model completely collapses in terms of temporal grounding. The ablated one only temporally grounds a small fragment, about 1.5 seconds, of the full event with a much smaller IoU of 0.11. Such a difference points out the utmost importance of the propagation module for

temporal grounding. The whole base model can place simply one point of the anomaly; however, with the propagation, the agent can track the duration of the glitch states. This result demonstrates how propagation transforms the task from an open-ended detection problem into a targeted verification query, enabling complete and accurate segmentation of the entire glitch event.

5.2.3 Discussion

Limitations in Architecture

Several limitations face the current method despite unseen performance gains we have demonstrated in our experiments. Firstly, video segmentation is carried out such that there is a fixed window size, w_s: with this construction, the GlitchAgent only analyses and outputs detections in discrete local windows. Thus, it cannot accurately perceive temporal dynamics that occur across the boundaries of two windows. For example, a glitch event initiated by interference in the last few frames of one window and finishing in the first few frames of another window may well be missed by the GlitchAgent. Secondly, while the propagation mechanism works well in the duration propagation of glitches, it can only operate at a window level rather than a more desirable frame level. Its coarse granularity places an inherent bottleneck on its performance, as shown in our experimental results where the mIoU score stagnates at around 0.55. The GlitchAgent results in Table 5.4 illustrate these two limitations. First, GlitchAgent fails to detect a minor event, "After the player kills the Triceratops." This is due to the fact that the event lies precisely at the boundary of two consecutive analysis windows. Second, GlitchAgent's temporal precision is limited by its window-based analysis: the agent can localize temporally only as far as the start of the window (0s), so it cannot resolve to the more precise ground truth start time (2s).

Despite the aforementioned issues, optimizing glitch detection under the premise of training-free is better achieved using a window-based method on current MLLMs than inputting all frames of a complete video simultaneously.

Limitations in Annotation

Creating a ground-truth dataset for game glitches through manual annotation of gameplay videos suffers from some inherent limitations in terms of glitch description and temporal localization. A primary issue concerns the reliance on mere visual data that limits truly logic-based glitches. Anomalies stemming from internal states of the game, such as incorrect inventory calculation or flawed quest logic, frequently do not have salient visual correlates. Therefore, they are often very difficult for human annotators to label and almost impossible for any learning-based system founded on vision to identify, thus resulting in the underrepresentation of significant glitch categories. In addition to that, temporal boundary annotation (start and end timestamps) is subject to the impreciseness and subjectivity of the annotator. Inter-annotator disagreement remains quite significant when it comes to the exact span of the causal chain, even when a predefined set of guidelines has been laid down that requires the annotators to specify what causes and what effects a given glitch. This means one can guarantee the short manifestation of the glitch is definitely captured, but one has a very coarse assumption of the time accuracy largescale around seconds, which introduces ambiguity in the downstream training and evaluation of glitch detection systems.

Similar Task: Anomaly Detection

The proposed task of detecting game glitches in videos shares with anomaly detection in videos some basic fundamental concepts while posing different challenges. At their core,

they are all temporal localization problems for atypical events that deviate from a normative pattern. Both require models not merely to identify an atypical event but also to provide precise timestamp entries for the commencement and termination of the event within a longer video sequence. Besides, both require some degree of semantic awareness to sift between anomalies that do have significance and mere trivial variations. However, the core difference stems from the scope and cardinality of the solution. Very often, a conventional anomaly detector is laid out as a single-instance localization problem, thereby single-mindedly searching for one primary anomalous event, like a car crash, to be able to conclusively describe it. Our glitch detection, on the other hand, is a multiple-instance problem that requires a system to find and describe each glitch in a video exhaustively as a single gameplay session can contain many different, unintended behaviors. Another fact is that video game glitches can be more complex and difficult to be detected, since anomalies can be detected by single frames, glitches need temporal context. In addition, current state-of-the-art methods in anomaly detection heavily rely on fine-tuning with large-scale well-annotated datasets, which is difficult to apply to the glitch detection task under the current lack of dataset.

Chapter 6

Conclusions

This thesis takes on the monumental challenge of automating QA in the video game industry: a process that presently suffers from being slow, expensive, and incompletely manual. It is assumed that detecting video game glitches from gameplay footage is a complicated task: it lacks annotated data, needs temporal context for its identification, and in practice requires pinpoint temporal localization. To solve this, the work presents a solution consisting of a new dataset, an elaborate detection framework, and a more rigorous metric to push the state-of-the-art in automatic video glitch detection.

With the conclusion, the main contributions and highlights of this research are briefly summarized. We will outline the implications this work holds for game development and automated testing in the future while being rather critical of the limitations of our approach and introducing several interesting avenues for future work.

6.1 Summary of Key Findings

The primary contributions of this thesis are threefold, each designed to tackle a specific challenge in the field. Our experimental results have validated the effectiveness of our approach, leading to several key findings:

- **The GlitchAgent framework significantly enhances MLLM performance for**

glitch detection. Our experiments demonstrated that by decomposing the monolithic task of analyzing a long video into a multi-stage process—window-based hypothesis generation, temporal propagation, and canonical description synthesis—we can substantially improve the performance of various MLLM backbones. Specifically, our method consistently boosts detection precision by mitigating model hallucination and dramatically improves temporal grounding accuracy (mIoU) compared to naive, end-to-end prompting on the same models.

- **Temporal propagation is essential for accurate localization.** Our ablation studies revealed that the temporal propagation mechanism—using verified glitch "anchors" to trace the full duration of an event—is a cornerstone of our framework's success. Disabling this component resulted in a catastrophic drop in temporal grounding performance across all tested models, confirming that an isolated, window-by-window analysis is insufficient for capturing the complete timeline of a glitch. Our method effectively bridges these temporal gaps to produce a contiguous and accurate event timeline.

- **The LLM-Judged Glitch Detection Score (GDS) offers a more human-aligned evaluation.** We identified critical flaws in traditional metrics like SODA when applied to glitch detection, namely their reliance on lexical similarity (e.g., METEOR) and temporally-greedy matching. Our proposed GDS metric, which uses an LLM-as-judge for semantic scoring and a globally optimal matching algorithm, proved to be far more robust. It showed a significantly higher correlation with human judgment, providing a more reliable and nuanced assessment of a system's ability to generate semantically correct and temporally precise glitch reports.

- **The VideoGlitch dataset provides a new, challenging benchmark for the community.** To facilitate this research, we created and introduced VideoGlitch,

the first large-scale dataset for open-ended glitch detection featuring detailed, natural language descriptions and precise temporal annotations. Its creation through a semi-automated pipeline demonstrates a scalable method for generating high-quality data in specialized domains. This dataset will enable future research and provide a standardized benchmark for comparing different approaches.

6.2 Limitations

Despite its promising results, this work is not without its limitations, which offer clear avenues for future improvement:

- **Dependence on Backbone MLLM Capabilities:** The performance of GlitchAgent is fundamentally bound by the visual and reasoning capabilities of its underlying MLLM. While our framework improves the performance of all tested models, it cannot create knowledge that the backbone model does not possess. Errors in the MLLM's core understanding can still propagate through the system.
- **Computational Expense:** The multi-stage nature of GlitchAgent, which involves multiple LLM inference calls for a single video (hypothesis generation, propagation, and synthesis), is computationally expensive and slow. It is not suitable for real-time analysis in its current form and may be cost-prohibitive for very large-scale deployment using proprietary model APIs.

6.3 Future Research Directions

Based on the findings of this thesis, we propose the following avenues for future research:

- **Development of Specialized and Efficient Models:** A plausible next step could be training more efficient and better model using fine-tuning on the high-quality, well-structured data. This model should focus on improving the understandings of temporal dynamics.

- **Multimodal Glitch Detection:** Future work should go beyond video frames to consider other data streams. Taking the auditory modality to spot sound-related glitches should be pursued. The fusion of the system with performance telemetry issued by the game engine-that is, frame rate logs, memory usage, and console error outputs-would enable it to consider performance and technical glitches-right now out of its range.

- **Direct Game Engine Integration:** A long-term vision for this work is to move beyond passive video analysis and instead integrate a similar agent directly into the game engine (through, for example, a plugin for Unreal Engine or Unity). Such an agent would not be limited to only working with the visual outputs, but instead, would be able to consider the underlying game state-object positions, character health, quest flags, and so forth. This would enable the detection of an entire class of logical and non-visual bugs with near-perfect accuracy.

- **Interactive and Exploratory Agents:** Finally, research may shift from passive detection to active testing. An autonomous agent might be developed that not only detects glitches but also actively tries to cause them. Guided by a detection model such as GlitchAgent, this agent could learn to explore game mechanics and environments most likely to expose hidden errors, bringing it closer to a truly automated and intelligent QA partner.

Bibliography

[1] Amazon AGI, Aaron Langford, Aayush Shah, Abhanshu Gupta, Abhimanyu Bhatter, Abhinav Goyal, Abhinav Mathur, Abhinav Mohanty, Abhishek Kumar, Abhishek Sethi, Abi Komma, Abner Pena, Achin Jain, Adam Kunysz, Adam Opyrchal, Adarsh Singh, Aditya Rawal, Adok Achar Budihal Prasad, Adrià de Gispert, Agnika Kumar, Aishwarya Aryamane, Ajay Nair, Akilan M, Akshaya Iyengar, Akshaya Vishnu Kudlu Shanbhogue, Alan He, Alessandra Cervone, Alex Loeb, Alex Zhang, Alexander Fu, Alexander Lisnichenko, Alexander Zhipa, Alexandros Potamianos, Ali Kebarighotbi, Aliakbar Daronkolaei, Alok Parmesh, Amanjot Kaur Samra, Ameen Khan, Amer Rez, Amir Saffari, Amit Agarwalla, Amit Jhindal, Amith Mamidala, Ammar Asmro, Amulya Ballakur, Anand Mishra, Anand Sridharan, Anastasiia Dubinina, Andre Lenz, Andreas Doerr, Andrew Keating, Andrew Leaver, Andrew Smith, Andrew Wirth, Andy Davey, Andy Rosenbaum, Andy Sohn, Angela Chan, Aniket Chakrabarti, Anil Ramakrishna, Anirban Roy, Anita Iyer, Anjali Narayan-Chen, Ankith Yennu, Anna Dabrowska, Anna Gawlowska, Anna Rumshisky, Anna Turek, Anoop Deoras, Anton Bezruchkin, Anup Prasad, Anupam Dewan, Anwith Kiran, Apoorv Gupta, Aram Galstyan, Aravind Manoharan, Arijit Biswas, Arindam Mandal, Arpit Gupta, Arsamkhan Pathan, Arun Nagarajan, Arushan Rajasekaram, Arvind Sundararajan, Ashwin Ganesan, Ashwin Swaminathan, Athanasios Mouchtaris, Audrey Champeau, Avik Ray, Ayush Jaiswal, Ayush Sharma, Bailey Keefer, Balamurugan Muthiah, Beatriz Leon-Millan, Ben Koopman, Ben Li, Benjamin Biggs, Benjamin Ott, Bhanu Vinzamuri, Bharath Venkatesh, Bhavana Ganesh, Bhoomit Vasani, Bill Byrne, Bill Hsu, Bincheng Wang, Blake King, Blazej Gorny, Bo Feng, Bo Zheng, Bodhisattwa Paul, Bofan Sun, Bofeng Luo, Bowen

Chen, Bowen Xie, Boya Yu, Brendan Jugan, Brett Panosh, Brian Collins, Brian Thompson, Can Karakus, Can Liu, Carl Lambrecht, Carly Lin, Carolyn Wang, Carrie Yuan, Casey Loyda, Cezary Walczak, Chalapathi Choppa, Chandana Satya Prakash, Chankrisna Richy Meas, Charith Peris, Charles Recaido, Charlie Xu, Charul Sharma, Chase Kernan, Chayut Thanapirom, Chengwei Su, Chenhao Xu, Chenhao Yin, Chentao Ye, Chenyang Tao, Chethan Parameshwara, Ching-Yun Chang, Chong Li, Chris Hench, Chris Tran, Christophe Dupuy, Christopher Davis, Christopher DiPersio, Christos Christodoulopoulos, Christy Li, Chun Chen, Claudio Delli Bovi, Clement Chung, Cole Hawkins, Connor Harris, Corey Ropell, Cynthia He, DK Joo, Dae Yon Hwang, Dan Rosen, Daniel Elkind, Daniel Pressel, Daniel Zhang, Danielle Kimball, Daniil Sorokin, Dave Goodell, Davide Modolo, Dawei Zhu, Deepikaa Suresh, Deepti Ragha, Denis Filimonov, Denis Foo Kune, Denis Romasanta Rodriguez, Devamanyu Hazarika, Dhananjay Ram, Dhawal Parkar, Dhawal Patel, Dhwanil Desai, Dinesh Singh Rajput, Disha Sule, Diwakar Singh, Dmitriy Genzel, Dolly Goldenberg, Dongyi He, Dumitru Hanciu, Dushan Tharmal, Dzmitry Siankovich, Edi Cikovic, Edwin Abraham, Ekraam Sabir, Elliott Olson, Emmett Steven, Emre Barut, Eric Jackson, Ethan Wu, Evelyn Chen, Ezhilan Mahalingam, Fabian Triefenbach, Fan Yang, Fangyu Liu, Fanzi Wu, Faraz Tavakoli, Farhad Khozeimeh, Feiyang Niu, Felix Hieber, Feng Li, Firat Elbey, Florian Krebs, Florian Saupe, Florian Sprünken, Frank Fan, Furqan Khan, Gabriela De Vincenzo, Gagandeep Kang, George Ding, George He, George Yeung, Ghada Qaddoumi, Giannis Karamanolakis, Goeric Huybrechts, Gokul Maddali, Gonzalo Iglesias, Gordon McShane, Gozde Sahin, Guangtai Huang, Gukyeong Kwon, Gunnar A. Sigurdsson, Gurpreet Chadha, Gururaj Kosuru, Hagen Fuerstenau, Hah Hah, Haja Maideen, Hajime Hosokawa, Han Liu, Han-Kai Hsu, Hann Wang, Hao Li, Hao Yang, Haofeng Zhu, Haozheng Fan, Harman Singh, Harshavardhan Kaluvala, Hashim Saeed, He Xie, Helian Feng, Hendrix Luo, Hengzhi Pei, Henrik Nielsen, Hesam Ilati, Himanshu Patel, Hong-

shan Li, Hongzhou Lin, Hussain Raza, Ian Cullinan, Imre Kiss, Inbarasan Thangamani, Indrayani Fadnavis, Ionut Teodor Sorodoc, Irem Ertuerk, Iryna Yemialyanava, Ishan Soni, Ismail Jelal, Ivan Tse, Jack FitzGerald, Jack Zhao, Jackson Rothgeb, Jacky Lee, Jake Jung, Jakub Debski, Jakub Tomczak, James Jeun, James Sanders, Jason Crowley, Jay Lee, Jayakrishna Anvesh Paidy, Jayant Tiwari, Jean Farmer, Jeff Solinsky, Jenna Lau, Jeremy Savareese, Jerzy Zagorski, Ji Dai, Jiacheng, Gu, Jiahui Li, Jian, Zheng, Jianhua Lu, Jianhua Wang, Jiawei Dai, Jiawei Mo, Jiaxi Xu, Jie Liang, Jie Yang, Jim Logan, Jimit Majmudar, Jing Liu, Jinghong Miao, Jingru Yi, Jingyang Jin, Jiun-Yu Kao, Jixuan Wang, Jiyang Wang, Joe Pemberton, Joel Carlson, Joey Blundell, John Chin-Jew, John He, Jonathan Ho, Jonathan Hueser, Jonathan Lunt, Jooyoung Lee, Joshua Tan, Joyjit Chatterjee, Judith Gaspers, Jue Wang, Jun Fang, Jun Tang, Jun Wan, Jun Wu, Junlei Wang, Junyi Shi, Justin Chiu, Justin Satriano, Justin Yee, Jwala Dhamala, Jyoti Bansal, Kai Zhen, Kai-Wei Chang, Kaixiang Lin, Kalyan Raman, Kanthashree Mysore Sathyendra, Karabo Moroe, Karan Bhandarkar, Karan Kothari, Karolina Owczarzak, Karthick Gopalswamy, Karthick Ravi, Karthik Ramakrishnan, Karthika Arumugam, Kartik Mehta, Katarzyna Konczalska, Kavya Ravikumar, Ke Tran, Kechen Qin, Kelin Li, Kelvin Li, Ketan Kulkarni, Kevin Angelo Rodrigues, Keyur Patel, Khadige Abboud, Kiana Hajebi, Klaus Reiter, Kris Schultz, Krishna Anisetty, Krishna Kotnana, Kristen Li, Kruthi Channamallikarjuna, Krzysztof Jakubczyk, Kuba Pierewoj, Kunal Pal, Kunwar Srivastav, Kyle Bannerman, Lahari Poddar, Lakshmi Prasad, Larry Tseng, Laxmikant Naik, Leena Chennuru Vankadara, Lenon Minorics, Leo Liu, Leonard Lausen, Leonardo F. R. Ribeiro, Li Zhang, Lili Gehorsam, Ling Qi, Lisa Bauer, Lori Knapp, Lu Zeng, Lucas Tong, Lulu Wong, Luoxin Chen, Maciej Rudnicki, Mahdi Namazifar, Mahesh Jaliminche, Maira Ladeira Tanke, Manasi Gupta, Mandeep Ahlawat, Mani Khanuja, Mani Sundaram, Marcin Leyk, Mariusz Momotko, Markus Boese, Markus Dreyer, Markus Mueller, Mason Fu, Mateusz

Górski, Mateusz Mastalerczyk, Matias Mora, Matt Johnson, Matt Scott, Matthew Wen, Max Barysau, Maya Boumerdassi, Maya Krishnan, Mayank Gupta, Mayank Hirani, Mayank Kulkarni, Meganathan Narayanasamy, Melanie Bradford, Melanie Gens, Melissa Burke, Meng Jin, Miao Chen, Michael Denkowski, Michael Heymel, Michael Krestyaninov, Michal Obirek, Michalina Wichorowska, Michał Miotk, Milosz Watroba, Mingyi Hong, Mingzhi Yu, Miranda Liu, Mohamed Gouda, Mohammad El-Shabani, Mohammad Ghavamzadeh, Mohit Bansal, Morteza Ziyadi, Nan Xia, Nathan Susanj, Nav Bhasin, Neha Goswami, Nehal Belgamwar, Nicolas Anastassacos, Nicolas Bergeron, Nidhi Jain, Nihal Jain, Niharika Chopparapu, Nik Xu, Nikko Strom, Nikolaos Malandrakis, Nimisha Mishra, Ninad Parkhi, Ninareh Mehrabi, Nishita Sant, Nishtha Gupta, Nitesh Sekhar, Nithin Rajeev, Nithish Raja Chidambaram, Nitish Dhar, Noor Bhagwagar, Noy Konforty, Omar Babu, Omid Razavi, Orchid Majumder, Osama Dar, Oscar Hsu, Pablo Kvitca, Pallavi Pandey, Parker Seegmiller, Patrick Lange, Paul Ferraro, Payal Motwani, Pegah Kharazmi, Pei Wang, Pengfei Liu, Peter Bradtke, Peter Götz, Peter Zhou, Pichao Wang, Piotr Poskart, Pooja Sonawane, Pradeep Natarajan, Pradyun Ramadorai, Pralam Shah, Prasad Nirantar, Prasanthi Chavali, Prashan Wanigasekara, Prashant Saraf, Prashun Dey, Pratyush Pant, Prerak Pradhan, Preyaa Patel, Priyanka Dadlani, Prudhvee Narasimha Sadha, Qi Dong, Qian Hu, Qiaozi, Gao, Qing Liu, Quinn Lam, Quynh Do, R. Manmatha, Rachel Willis, Rafael Liu, Rafal Ellert, Rafal Kalinski, Rafi Al Attrach, Ragha Prasad, Ragini Prasad, Raguvir Kunani, Rahul Gupta, Rahul Sharma, Rahul Tewari, Rajaganesh Baskaran, Rajan Singh, Rajiv Gupta, Rajiv Reddy, Rajshekhar Das, Rakesh Chada, Rakesh Vaideeswaran Mahesh, Ram Chandrasekaran, Ramesh Nallapati, Ran Xue, Rashmi Gangadharaiah, Ravi Rachakonda, Renxian Zhang, Rexhina Blloshmi, Rishabh Agrawal, Robert Enyedi, Robert Lowe, Robik Shrestha, Robinson Piramuthu, Rohail Asad, Rohan Khanna, Rohan Mukherjee, Rohit Mittal, Rohit Prasad, Rohith Mysore Vijaya Kumar, Ron Diamant, Ruchita

Gupta, Ruiwen Li, Ruoying Li, Rushabh Fegade, Ruxu Zhang, Ryan Arbow, Ryan Chen, Ryan Gabbard, Ryan Hoium, Ryan King, Sabarishkumar Iyer, Sachal Malick, Sahar Movaghati, Sai Balakavi, Sai Jakka, Sai Kashyap Paruvelli, Sai Muralidhar Jayanthi, Saicharan Shriram Mujumdar, Sainyam Kapoor, Sajjad Beygi, Saket Dingliwal, Saleh Soltan, Sam Ricklin, Sam Tucker, Sameer Sinha, Samridhi Choudhary, Samson Tan, Samuel Broscheit, Samuel Schulter, Sanchit Agarwal, Sandeep Atluri, Sander Valstar, Sanjana Shankar, Sanyukta Sanyukta, Sarthak Khanna, Sarvpriye Khetrapal, Satish Janakiraman, Saumil Shah, Saurabh Akolkar, Saurabh Giri, Saurabh Khandelwal, Saurabh Pawar, Saurabh Sahu, Sean Huang, Sejun Ra, Senthilkumar Gopal, Sergei Dobroshinsky, Shadi Saba, Shamik Roy, Shamit Lal, Shankar Ananthakrishnan, Sharon Li, Shashwat Srijan, Shekhar Bhide, Sheng Long Tang, Sheng Zha, Shereen Oraby, Sherif Mostafa, Shiqi Li, Shishir Bharathi, Shivam Prakash, Shiyuan Huang, Shreya Yembarwar, Shreyas Pansare, Shreyas Subramanian, Shrijeet Joshi, Shuai Liu, Shuai Tang, Shubham Chandak, Shubham Garg, Shubham Katiyar, Shubham Mehta, Shubham Srivastav, Shuo Yang, Siddalingesha D S, Siddharth Choudhary, Siddharth Singh Senger, Simon Babb, Sina Moeini, Siqi Deng, Siva Loganathan, Slawomir Domagala, Sneha Narkar, Sneha Wadhwa, Songyang Zhang, Songyao Jiang, Sony Trenous, Soumajyoti Sarkar, Soumya Saha, Sourabh Reddy, Sourav Dokania, Spurthideepika Sandiri, Spyros Matsoukas, Sravan Bodapati, Sri Harsha Reddy Wdaru, Sridevi Yagati Venkateshdatta, Srikanth Ronanki, Srinivasan R Veeravanallur, Sriram Venkatapathy, Sriramprabhu Sankaraguru, Sruthi Gorantla, Sruthi Karuturi, Stefan Schroedl, Subendhu Rongali, Subhasis Kundu, Suhaila Shakiah, Sukriti Tiwari, Sumit Bharti, Sumita Sami, Sumith Mathew, Sunny Yu, Sunwoo Kim, Suraj Bajirao Malode, Susana Cumplido Riel, Swapnil Palod, Swastik Roy, Syed Furqhan, Tagyoung Chung, Takuma Yoshitani, Taojiannan Yang, Tejaswi Chillakura, Tejwant Bajwa, Temi Lajumoke, Thanh Tran, Thomas Gueudre, Thomas Jung, Tianhui Li, Tim Seemman, Timothy Leffel, Tingting Xiang, Tirth

Patel, Tobias Domhan, Tobias Falke, Toby Guo, Tom Li, Tomasz Horszczaruk, Tomasz Jedynak, Tushar Kulkarni, Tyst Marin, Tytus Metrycki, Tzu-Yen Wang, Umang Jain, Upendra Singh, Utkarsh Chirimar, Vaibhav Gupta, Vanshil Shah, Varad Deshpande, Varad Gunjal, Varsha Srikeshava, Varsha Vivek, Varun Bharadwaj, Varun Gangal, Varun Kumar, Venkatesh Elango, Vicente Ordonez, Victor Soto, Vignesh Radhakrishnan, Vihang Patel, Vikram Singh, Vinay Varma Kolanuvada, Vinayshekhar Bannihatti Kumar, Vincent Auvray, Vincent Cartillier, Vincent Ponzo, Violet Peng, Vishal Khandelwal, Vishal Naik, Vishvesh Sahasrabudhe, Vitaliy Korolev, Vivek Gokuladas, Vivek Madan, Vivek Subramanian, Volkan Cevher, Vrinda Gupta, Wael Hamza, Wei Zhang, Weitong Ruan, Weiwei Cheng, Wen Zhang, Wenbo Zhao, Wenyan Yao, Wenzhuo Ouyang, Wesley Dashner, William Campbell, William Lin, Willian Martin, Wyatt Pearson, Xiang Jiang, Xiangxing Lu, Xiangyang Shi, Xianwen Peng, Xiaofeng Gao, Xiaoge Jiang, Xiaohan Fei, Xiaohui Wang, Xiaozhou Joey Zhou, Xin Feng, Xinyan Zhao, Xinyao Wang, Xinyu Li, Xu Zhang, Xuan Wang, Xuandi Fu, Xueling Yuan, Xuning Wang, Yadunandana Rao, Yair Tavizon, Yan Rossiytsev, Yanbei Chen, Yang Liu, Yang Zou, Yangsook Park, Yannick Versley, Yanyan Zhang, Yash Patel, Yen-Cheng Lu, Yi Pan, Yi-Hsiang, Lai, Yichen Hu, Yida Wang, Yiheng Zhou, Yilin Xiang, Ying Shi, Ying Wang, Yishai Galatzer, Yongxin Wang, Yorick Shen, Yuchen Sun, Yudi Purwatama, Yue, Wu, Yue Gu, Yuechun Wang, Yujun Zeng, Yuncong Chen, Yunke Zhou, Yusheng Xie, Yvon Guy, Zbigniew Ambrozinski, Zhaowei Cai, Zhen Zhang, Zheng Wang, Zhenghui Jin, Zhewei Zhao, Zhiheng Li, Zhiheng Luo, Zhikang Zhang, Zhilin Fang, Zhiqi Bu, Zhiyuan Wang, Zhizhong Li, Zijian Wang, Zimeng, Qiu, and Zishi Li. The amazon nova family of models: Technical report and model card, 2025. URL https://arxiv.org/abs/2506.12103.

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana

Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

[3] Anthropic. Introducing computer use, a new claude 3.5 sonnet, and claude 3.5 haiku. https://seed-tars.com/1.5, 2024.

[4] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

[5] Meng Cao, Haoran Tang, Haoze Zhao, Hangyu Guo, Jiaheng Liu, Ge Zhang, Ruyang Liu, Qiang Sun, Ian Reid, and Xiaodan Liang. Physgame: Uncovering physical commonsense violations in gameplay videos. *ArXiv*, abs/2412.01800, 2024. URL https://api.semanticscholar.org/CorpusID:274437775.

[6] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, Lixin Gu, Xuehui Wang, Qingyun Li, Yimin Ren, Zixuan Chen, Jiapeng Luo, Jiahao Wang, Tan Jiang, Bo Wang, Conghui He, Botian Shi, Xingcheng Zhang, Han Lv, Yi Wang, Wenqi Shao, Pei Chu, Zhongying Tu, Tong He, Zhiyong Wu, Huipeng Deng, Jiaye Ge, Kai Chen, Kaipeng Zhang, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling, 2025. URL https://arxiv.org/abs/2412.05271.

[7] Soichiro Fujita, Tsutomu Hirao, Hidetaka Kamigaito, Manabu Okumura, and Masaaki Nagata. Soda: Story oriented dense video captioning evaluation flamework. In *Proceedings of the European Conference on Computer Vision (ECCV)*, August 2020.

[8] Google. Introducing gemini 2.0: our new ai model for the agentic era. Technical report, Google, 2024. URL https://blog.google/technology/google-deepmind/google-gemini-ai-update-december-2024/.

[9] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Peiyuan Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. Llava-onevision: Easy visual task transfer, 2024. URL https://arxiv.org/abs/2408.03326.

[10] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *International conference on machine learning*, pages 19730–19742. PMLR, 2023.

[11] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916, 2023.

[12] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 26296–26306, 2024.

[13] 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.

[14] OpenAI. Gpt-4o mini: advancing cost-efficient intelligence. Technical report, OpenAI, 2024. URL https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/.

[15] ByteDance Seed. Ui-tars-1.5. https://seed-tars.com/1.5, 2025.

[16] Mohammad Reza Taesiri, Finlay Macklon, and Cor-Paul Bezemer. Clip meets game-physics: Towards bug identification in gameplay videos using zero-shot transfer learning, 2022. URL https://arxiv.org/abs/2203.11096.

[17] Mohammad Reza Taesiri, Finlay Macklon, Yihe Wang, Hengshuo Shen, and Cor-Paul Bezemer. Large language models are pretty good zero-shot video game bug detectors. *ArXiv*, abs/2210.02506, 2022. URL https://api.semanticscholar.org/CorpusID:252735080.

[18] Mohammad Reza Taesiri, Tianjun Feng, Cor-Paul Bezemer, and Anh Totti Nguyen. Glitchbench: Can large multimodal models detect video game glitches? *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 22444–22455, 2023. URL https://api.semanticscholar.org/CorpusID:266163155.

[19] Kimi Team, Angang Du, Bohong Yin, Bowei Xing, Bowen Qu, Bowen Wang, Cheng Chen, Chenlin Zhang, Chenzhuang Du, Chu Wei, Congcong Wang, Dehao Zhang, Dikang Du, Dongliang Wang, Enming Yuan, Enzhe Lu, Fang Li, Flood Sung, Guangda Wei, Guokun Lai, Han Zhu, Hao Ding, Hao Hu, Hao Yang, Hao Zhang, Haoning Wu, Haotian Yao, Haoyu Lu, Heng Wang, Hongcheng Gao, Huabin Zheng, Jiaming Li, Jianlin Su, Jianzhou Wang, Jiaqi Deng, Jiezhong Qiu, Jin Xie, Jinhong Wang, Jingyuan Liu, Junjie Yan, Kun Ouyang, Liang Chen, Lin Sui, Longhui Yu, Mengfan Dong, Mengnan Dong, Nuo Xu, Pengyu Cheng, Qizheng Gu, Runjie Zhou, Shaowei Liu, Sihan Cao, Tao Yu, Tianhui Song, Tongtong Bai, Wei Song, Weiran He, Weixiao Huang, Weixin Xu, Xiaokun Yuan, Xingcheng Yao, Xingzhe Wu, Xinhao Li, Xinxing Zu, Xinyu Zhou, Xinyuan Wang, Y. Charles, Yan Zhong, Yang Li, Yangyang Hu, Yanru Chen, Yejie Wang, Yibo Liu, Yibo Miao, Yidao Qin, Yimin Chen, Yiping Bao, Yiqin Wang, Yong-

sheng Kang, Yuanxin Liu, Yuhao Dong, Yulun Du, Yuxin Wu, Yuzhi Wang, Yuzi Yan, Zaida Zhou, Zhaowei Li, Zhejun Jiang, Zheng Zhang, Zhilin Yang, Zhiqi Huang, Zihao Huang, Zijia Zhao, Ziwei Chen, and Zongyu Lin. Kimi-vl technical report, 2025. URL https://arxiv.org/abs/2504.07491.

[20] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *Advances in neural information processing systems*, 35:10078–10093, 2022.

[21] Yi Wang, Xinhao Li, Ziang Yan, Yinan He, Jiashuo Yu, Xiangyu Zeng, Chenting Wang, Changlian Ma, Haian Huang, Jianfei Gao, et al. Internvideo2. 5: Empowering video mllms with long and rich context modeling. *arXiv preprint arXiv:2501.12386*, 2025.

[22] Hang Zhang, Xin Li, and Lidong Bing. Video-llama: An instruction-tuned audio-visual language model for video understanding. *arXiv preprint arXiv:2306.02858*, 2023.

Appendices

Appendix A

Dataset Details

A.1 Statistics

Table A.1: Number of Videos by Genre and Sub-Genre

Genre	**Video Num**	**Sub-Genre**	**Video Num**
Action	530	Survival	70
		Platform	69
		Action adventure	68
		Shooter	69
		Battle royale	78
		Stealth	69
		Fighting	66
		Rhythm	11
		Wargame	30
Simulation	259	Vehicle simulation	67
		Construction & management	66
		Sports	65
		Life simulation	61
RPG	116	Action RPG	69
		Dungeon crawl	20
		Roguelike	8
		MMORPG	19
Adventure	10	Interactive film	10
Puzzle	43	-	43
Strategy	52	Real-time tactics (RTT)	28
		Turn-based strategy	24

Table A.2: Game Genres and Video Numbers

Genre	Sub-Genre	Game Name	Video Num
Action	Survival	ARK - Survival Evolved	10
		DayZ	10
		Dying Light	10
		Sony The Last of Us Series	10
		State of Decay 2	10
		Subnautica	10
		Wreckfest	10
	Platform	A Hat in Time	10
		Clustertruck	10
		Fall Guys - Ultimate Knockout	10
		Roblox	10
		Super Bunny Man	10
		Super Mario Sunshine	10
		Trials Fusion	10
	Action adventure	Assassin's Creed Odyssey	10

Continued on next page

Genre	Sub-Genre	Game Name	Video Num
		Batman - Arkham Knight	10
		Grand Theft Auto V	10
		Just Cause 3	10
		Star Wars Jedi - Fallen Order	10
		The Legend of Zelda - Breath of the Wild	10
		Watch Dogs 2	10
	Shooter	Battlefield 4	10
		Call of Duty - Modern Warfare	10
		Counter-Strike - Global Offensive	10
		Destiny 2	10
		Far Cry 5	10
		Half-Life - Alyx	10
		Tom Clancy's Rainbow Six Siege	10
	Battle royale	Apex Legends	20
		Fortnite	20
		PUBG MOBILE	19

Continued on next page

Genre	Sub-Genre	Game Name	Video Num
		PlayerUnknown's Battlegrounds	10
		Z1 Battle Royale	10
	Stealth	Dishonored 2	20
		Hitman 2	20
		Metal Gear Solid V - The Phantom Pain	20
		The Greatest Penguin Heist of All Time	10
	Fighting	Chivalry - Medieval Warfare	10
		EA Sports UFC 2	10
		For Honor	10
		Gang Beasts	10
		Mordhau	10
		Paint the Town Red	10
		WWE 2K19	10
	Rhythm	Skeletal Dance Party	12
	Wargame	World of Tanks	30
	Vehicle simulation	BeamNG.drive	10
Simulation			

Continued on next page

Genre	Sub-Genre	Game Name	Video Num
		Euro Truck Simulator 2	10
		Forza Horizon 4	10
		Kerbal Space Program	10
		Star Citizen	10
		The Crew	10
		War Thunder	10
	Construction and management simulation	Astroneer	10
		Besiege	10
		Cities - Skylines	10
		Minecraft	10
		No Man's Sky	10
		Poly Bridge	10
		Space Engineers	10
	Sports	F1 2020	10
		FIFA 21	10
		NBA 2K20	10
		NHL 20	10
		Rocket League	10
		Skate 3	10
		Steep	10
	Life simulation	Farming Simulator	13

Continued on next page

Genre	Sub-Genre	Game Name	Video Num
		Farming Simulator 19	15
		Goat Simulator	18
		The Sims 4	20
RPG	Action RPG	Cyberpunk 2077	10
		Dark Souls III	10
		Dragon Age - Inquisition	10
		FINAL FANTASY XV	10
		Fallout 4	10
		The Elder Scrolls V - Skyrim	10
		The Witcher 3 - Wild Hunt	10
	Dungeon crawl	Exanima	20
	Roguelike	Noita	10
	MMORPG	World of Warcraft	19
Adventure	Interactive film	Detroit - Become Human	10
Puzzle		Human - Fall Flat	30
		Portal 2	17

Continued on next page

Genre	**Sub-Genre**	**Game Name**	**Video Num**
Strategy	Real-time tactics (RTT)	Totally Accurate Battle Simulator	30
	Turn-based strategy	XCOM 2	30

Appendix B

Prompts

B.1 Pseudo Annotation Prompt

Pseudo Annotation Prompt

You are an expert in game testing. You will use your professional knowledge to analyze unnatural phenomena that appear in game videos to determine whether there are any game glitches.

Context

These are some keyframes from a video that I want to upload. The indexes of these frames begin from 0.

```
{Contextual frame info here}
{Reddit context info here}
```

Objective

The video should already be divided into several segments; the given video is a segment of a longer video.

Each segment should include sampled keyframes from the video. The sampling rate is 3 frames per second.

First, attempt to describe the detailed information within the video segment, such as the game scene, game objects, game characters, etc.

Next, describe any abnormal phenomena; if you find any bugs, generate a bug or glitch description. Only the bug or glitch description is needed.

Constraint

Your output should contain a detailed video segment description and a bug or glitch description. If there is no bug or glitch, state clearly: "No bug or glitch found in this video segment."

Response

Your output should be a dictionary in JSON format. An example looks like this:

```
{
  "segment_description": "detailed description of the segment",
  "bug_description": "bug description"
}
```

B.2 LLM-Judged Glitch Detection Score Prompt

LLM-Judged Glitch Detection Score Prompt

You are an expert in video glitch analysis. Your task is to evaluate the quality of a predicted description based on the given ground truth glitch description, focusing on the accurate identification/detection of the glitch.

Ground Truth Description:

[gt]

Predicted Description:

[pred]

Instructions:

- **Rate the quality of the predicted description on a scale from 0 to 5**, using the following criteria:

- **0: No quality**. - The predicted description is completely unrelated to the ground truth. - The descriptions refer to entirely different events or glitches. - The predicted description is irrelevant or nonsensical in the context of the glitch.

- **1: Very low quality**. - The predicted description mentions some elements, but they are incorrect or irrelevant. - Key details about the glitch are missing or inaccurately described. - The descriptions have minimal overlap in content or context.

- **2: Low quality**. - The predicted description includes minor aspects related to the glitch but misses the main behavior. - Important elements are absent or significantly misrepresented. - The description has some relevant terms but lacks context or specificity.

- **3: Moderate quality**. - The predicted description identifies the general type of glitch but lacks significant details. - Some key behaviors or elements are mentioned,

but important specifics are missing. - The description captures part of the glitch but omits certain actions or effects.

- **4: High quality**. - The predicted description captures the main glitch and includes most key details. - Minor differences exist (e.g., general terms used instead of specific names), but the overall glitch is accurately described. - The description reflects the primary behaviors and impacts of the glitch on the game experience.

- **5: Nearly identical**. - The predicted description accurately identifies the glitch with all key details matching the ground truth. - There are no significant differences; both descriptions refer to the same event with similar specificity. - The description includes all important behaviors, entities involved, and impacts on gameplay.

- **Provide a brief reasoning before your answer**, focusing on:

- How well the predicted description matches the ground truth in identifying the glitch.

- Specific similarities or differences in the behaviors, entities involved, and details mentioned. - The impact of any missing or additional information on the accuracy of the glitch identification.

- **Output Format**:

Please provide your analysis and your rating in the following JSON format:

```
{
  "reasoning": "<brief reasoning of the evaluation>",
  "rating": <number from 0 to 5>
}
```