BUGXAI: AI POWERED GLITCH AND BUG DETECTION TOOL IN VIDEO GAMES

Authors: Dhruvin Suthar, Prem Modsing, Kundan Kadam

Affiliations: Nilkamal School Of Mathematics, Applied Statistics & Analytics, NMIMS

Mentor by: Dr. Yogesh Naik

Guided by: Nikhil Pawanikar

Abstract

As video games evolve in complexity, traditional manual testing methods increasingly struggle to keep up with the rapidly changing demands of quality assurance. This often results in longer development cycles, elevated costs, and a heightened risk of undetected glitches that can compromise both gameplay and overall user experience. Bugs stemming from issues in gameplay physics, visual artifacts, frame anomalies, and texture inconsistencies not only diminish game stability but also adversely affect player satisfaction. In response to these challenges, this research presents an AI-driven automated bug detection system that leverages advanced deep learning techniques. The system utilizes a YOLO-based object detection framework, complemented by optical flow analysis, frame differencing, and physics-based anomaly detection methods. By systematically analysing gameplay footage, the proposed system can identify and classifying a wide range of glitches, including floating characters, teleportation issues, speed inconsistencies, clipping errors, and frame drops. This multi-faceted approach significantly reduces the dependency on manual testing, providing a more consistent, accurate, and efficient method for bug detection. In doing so, it not only minimizes human error but also accelerates the debugging process, leading to a smoother development workflow. Ultimately, this research aims to optimize game quality assurance processes and enhance the overall production pipeline, ensuring that games meet the high standards expected by today's players. By integrating state-of-the-art deep learning techniques into automated testing frameworks, the proposed system sets a new benchmark for effective and scalable game testing solutions.

Keywords: *Game bug detection, AI-powered testing, deep learning, YOLO detection, gameplay anomaly detection, physics-based glitches.*

Introduction

The video game industry has seen remarkable advancements in graphics, physics engines, and artificial intelligence, pushing the boundaries of realism and interactivity. However, as games become more complex, ensuring a seamless, bug-free experience has become increasingly challenging. Bugs and glitches ranging from minor graphical errors to major game-breaking issues can severely impact player satisfaction, damage a game's reputation, and lead to financial losses for developers. Given the increasing sophistication of modern games and rising player expectations, effective and efficient game testing is now more crucial than ever. Traditionally, game testing relies on manual quality assurance (QA) teams that meticulously play through different scenarios to identify potential issues. While this approach has been the industry standard, it is time-consuming, costly, and often inefficient, especially for large-scale games. Moreover, human testers may miss certain glitches due to fatigue, variability in playstyles, or limitations in identifying subtle anomalies. To address these challenges, game developers are now exploring automation as a means to enhance testing efficiency, accelerate development cycles, and ensure more comprehensive bug detection.

This research introduces an AI-powered automated bug and glitch detection system designed to streamline the game testing process. Leveraging deep learning, computer vision, and physics-based analysis, the system analysis gameplay footage to detect anomalies and classify potential glitches with high accuracy. By integrating object detection, optical flow, frame difference analysis, and physics-based consistency checks, the system can identify a wide range of game-breaking issues, such as floating characters, teleportation glitches, speed inconsistencies, clipping problems, and frame anomalies. With the growing complexity of video games, AI-based testing solutions present a scalable and reliable alternative to traditional manual methods, ensuring a higher-quality gaming experience while minimizing post-release issues.

Literature Review

A Video Game Testing Method Utilizing Deep Learning

Mohammad Reza Taesiri, Moslem Habibi, MohammadAmin Fazli (2020). CSI Journal on Computer Science and Engineering, 17(2), 26-33.

This study introduces an automated testing method for video games using deep convolutional neural networks (CNNs) to detect graphical anomalies and compatibility issues across different runtime environments. The proposed approach follows a three-phase process: (1) local and cloud execution of test scenarios, (2) graphical error detection through CNN-based frame analysis, and (3) automated identification of rendering errors via a deep learning-based classification model. The DenseNet model used in this system achieved a 94% graphical error detection rate with 90% accuracy. While the methodology enhances efficiency in graphical bug detection, its dependency on human testers for initial gameplay inputs and the exclusion of performance testing (e.g., frame rate drops, input lag) remain key limitations. Future work should explore reinforcement learning for fully automated gameplay testing and real-time debugging mechanisms.

Automated Bug Finding in Video Games: A Case Study for Runtime Monitoring

Varvaressos, S., Lavoie, K., Gaboury, S., & Hallé, S. (2017). Computers in Entertainment, 15(1), 1.

This paper investigates runtime verification as a technique for automated bug detection in video games. By leveraging the game loop as a primary observation point, the system monitors runtime events against formal specifications to identify inconsistencies. The study evaluated six diverse video games and demonstrated the efficacy of this approach in detecting gameplay bugs, unexpected AI behaviors, and unintended game physics effects. Additionally, a tool named Replayr was developed to visually reconstruct game states for debugging purposes. Despite its advantages in reducing manual testing efforts, limitations include the complexity of defining temporal logic rules and the challenges in detecting AI-related bugs that do not violate formal specifications. Future advancements should focus on AI-driven behavior modeli ng and automated extraction of game properties to enhance monitoring capabilities.

Augmenting Automated Game Testing with Deep Reinforcement Learning

Bergdahl, J., Gordillo, C., Tollmar, K., & Gisslén, L. (2021). ArXiv preprint arXiv:2103.15819.

This study explores the integration of deep reinforcement learning (DRL) into automated game testing to enhance test coverage and identify exploits such as missing collision meshes. The methodology enables AI agents to autonomously navigate game environments, detect unintended behaviors, and evaluate game difficulty by measuring interaction time. Applied primarily to first-person shooter (FPS) games, the approach demonstrates superior adaptability compared to scripted AI-based testing. However, computational demands for training DRL agents present scalability concerns, and the lack of contextual awareness in AI agents may lead to false positives. Future research should optimize computational efficiency, incorporate human feedback loops for result validation, and extend the framework to cover combat AI and player interactions.

Automatic Bug Detection in Games Using LSTM Networks

Azizi, E., & Zaman, L. (2023). ArXiv preprint arXiv:2312.08418.

This paper proposes an LSTM-based anomaly detection framework for identifying perceptual and behavioral bugs in video games. By analyzing sequential video frames, the system detects gameplay anomalies and categorizes bug types using clustering techniques such as DBSCAN. The framework was tested on two FPS games—World of Bugs (WOB) and Echo+—demonstrating effective perceptual bug identification. Findings reveal a 74% homogeneity score in clustering bug types, which improved to 85% upon refining bug classification. Despite its promising results, the framework's scalability to larger datasets, memory constraints, and lack of real-time implementation remain challenges. Future enhancements should focus on integrating real-time monitoring capabilities, expanding dataset size, and incorporating depth and color information for improved accuracy.

Comparative Analysis of Existing Bug Detection Methods and BUGXAI's Novel Approach

Existing AI-based game bug detection tools primarily focus on visual analysis, often overlooking performance-related glitches like frame drops or screen freezes. BUGXAI enhances detection by analyzing both visual and performance metrics, ensuring a more comprehensive evaluation. Unlike traditional models, it identifies a broader range of issues, including Physics Glitches, Visual Glitches, Texture Glitches, and more.

Most bug detection tools rely on generic algorithms that fail to adapt to different game environments. BUGXAI leverages YOLOv8's deep learning capabilities, enabling it to learn from diverse datasets and detect bugs across various game genres. This adaptive approach ensures higher accuracy and relevance compared to conventional static detection methods. Standard bug detection methods provide limited feedback, often just highlighting anomalies without detailed analysis. BUGXAI, in contrast, offers in-depth logs, visualization tools, and actionable insights, helping developers not only detect but also understand the root causes of issues. By integrating precision-recall metrics and confusion matrices, it ensures better debugging efficiency.

BUGXAI's key innovation lies in its ability to process real-time video footage, analyzing frame-by-frame inconsistencies. It evaluates performance elements such as frame rate stability and unexpected freezes, setting a new benchmark in automated game testing. This real-time analysis provides game developers with immediate and actionable feedback.

By combining advanced AI with deep learning-driven bug classification, BUGXAI surpasses conventional detection methods. Its ability to analyze, categorize, and provide actionable insights on game glitches ensures a superior quality control process, making it an essential tool for modern game development.

BUGXAI: An AI-Powered Game Glitch and Bug Glitch Detection Tool

BUGXAI is an advanced AI-driven bug detection platform designed to address critical gaps in traditional game testing methods. Unlike conventional tools that rely solely on static rule-based detection, BUGXAI employs deep learning with YOLOv8 to analyze gameplay footage dynamically, ensuring a comprehensive and automated bug detection process. Unlike traditional systems that focus only on visual anomalies, BUGXAI integrates frame rate analysis, texture inconsistencies, physics irregularities, and screen freezes to identify a wide range of glitches. By leveraging real-time video processing and advanced machine learning, it provides game developers with precise, actionable insights beyond basic error detection.

BUGXAI generates detailed logs that highlight detected issues along with their probable causes, helping developers diagnose and fix bugs efficiently. This approach eliminates the shortcomings of existing solutions, such as limited adaptability, manual debugging inefficiencies, and lack of real-time performance analysis. Through its integration of AI-powered detection, performance tracking, and detailed visual feedback, BUGXAI redefines the standards for automated game testing. By improving efficiency, accuracy, and coverage in bug detection, it empowers game developers to deliver smoother, more polished gaming experiences with significantly reduced debugging time.

Data Collection Strategy

To ensure the robustness and diversity of the dataset used for training BUGXAI, our AI-powered glitch detection model, we implemented a multi-source data collection strategy. This approach helps in capturing a wide range of gameplay anomalies, enhancing the model's ability to identify and classify different types of glitches. The data collection process was carried out in the following steps:

1. Data Sources

To build a comprehensive dataset, three primary sources were utilized:

<u>YouTube Gameplay Videos</u> – Publicly available gameplay footage was gathered, focusing on graphical and physics-related anomalies across different games.

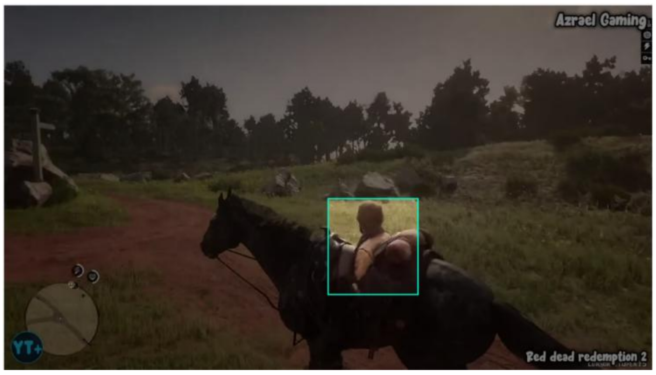
<u>Self-Collected Videos</u> – Custom gameplay recordings were made to ensure the inclusion of rare or underrepresented glitches that may not be commonly found in existing datasets.

<u>GameXPhysics Dataset</u> – A structured dataset containing physics-based glitches from various gaming environments was integrated to further enrich the dataset.

2. Data Extraction & Processing

Gameplay videos were converted into individual frames at a consistent frame rate to standardize the dataset. Frames were then analyzed to identify and extract glitches, ensuring only relevant instances were selected for further annotation.

3. Data Annotation & Categorization



Extracted frames were manually labeled into different categories based on the type of glitch detected through Roboflow software:

<u>Physics Glitch</u> – Includes errors related to object collisions, movement inconsistencies, gravity failures, and physics miscalculations.

<u>Visual Glitch</u> – Covers issues such as rendering artifacts, texture mismatches, missing assets, and UI-related anomalies.

<u>Frame Drop & Screen Freeze</u> – Identifies significant FPS drops and frozen frames, affecting gameplay smoothness.

The final dataset consists of 3600+ labeled images for model training and evaluation.

4. Dataset Preparation for Model Training

The labeled dataset was divided into:

<u>Training Set</u> (70%) – Used for model learning.

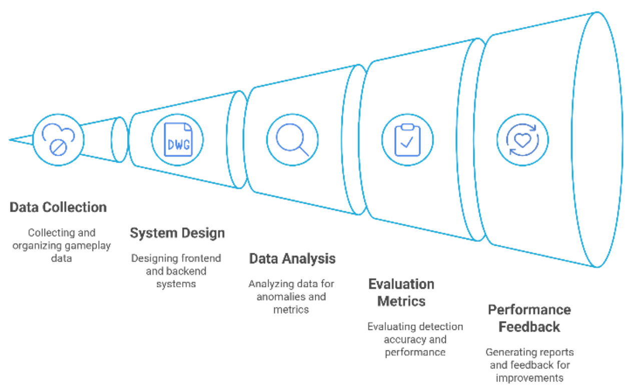
<u>Validation Set</u> (20%) – Helps in fine-tuning the model.

<u>Testing Set</u> (10%) – Evaluates the model’s final performance.

The data was formatted for compatibility with YOLOv8, ensuring efficient training and inference for real-time bug detection. This multi-source, well-structured dataset enhances BUGXAI’s ability to accurately detect, classify, and analyze video game glitches, making it a powerful tool for game developers and quality assurance teams.

Research Methodology

This research employs an applied methodology integrating machine learning, computer vision, and deep learning to develop an AI-driven bug detection system for video games. A quantitative and experimental approach is taken to evaluate the system’s efficiency in identifying and categorizing visual and physics-based glitches. The primary goal is to build a robust and scalable system capable of detecting anomalies in gameplay footage using automated AI techniques.


Data Collection
Collecting and organizing gameplay data
System Design
Designing frontend and backend systems
Data Analysis
Analyzing data for anomalies and metrics
Evaluation Metrics
Evaluating detection accuracy and performance
Performance Feedback
Generating reports and feedback for improvements

1. Data Collection Methods

The dataset used for training the glitch detection model is curated from multiple sources to ensure diversity and robustness. The data collection methodology includes:

<u>YouTube Gameplay Videos</u> – Publicly available gaming footage featuring various graphical and physics-based anomalies.

<u>Self-Collected Gameplay Data</u> – Custom gameplay recordings captured to include edge cases not well-represented in public datasets.

<u>Existing Datasets</u> – Incorporating structured datasets such as GameXPhysics, which provides structured examples of physics-based glitches in video games.

2. System Design and Development

The BUGXAI system consists of multiple functional layers that facilitate seamless processing and interaction:

<u>Frontend Layer (Streamlit UI)</u> – Provides an interface for users to upload videos, process them, visualize results, and download logs and processed videos.

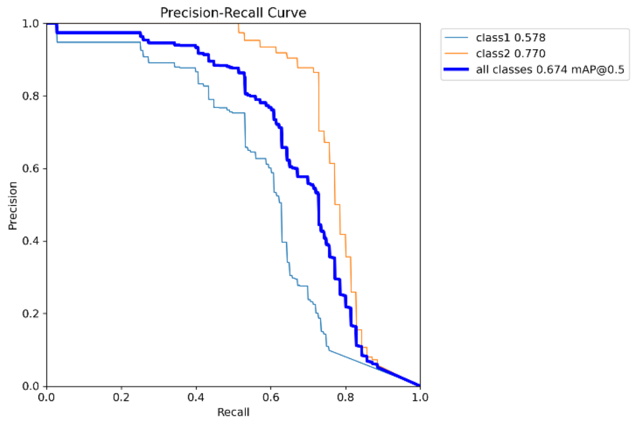
<u>Backend Layer (AI-Powered Processing)</u> – Utilizes YOLOv8 for real-time glitch detection, analyzing each frame for anomalies.

<u>Data Handling & Storage</u> – Logs are generated in real-time, allowing users to track detected glitches. The processed video and detailed logs can be downloaded in CSV or TXT format for further analysis.

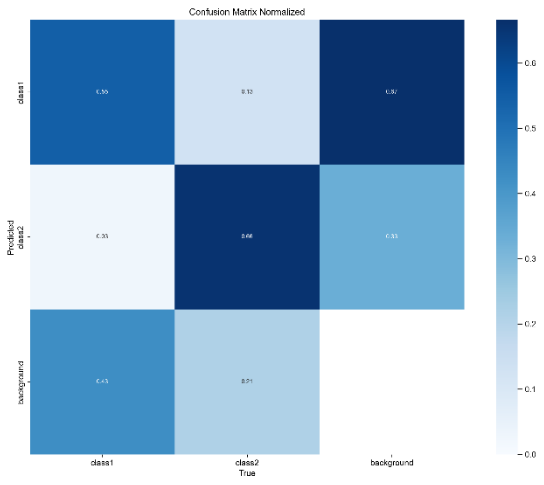
Data Analysis Approach

The processed video frames undergo object detection and anomaly classification using deep learning. The model is evaluated through key performance metrics such as:

<u>Precision & Recall</u> – To measure how effectively the model detects actual glitches.



<u>F1-Score & Confusion Matrices</u> – To analyze detection accuracy and false-positive/negative rates.



<u>Processing Speed</u> – Evaluating the system's real-time performance and efficiency.

Evaluation Metrics

The effectiveness of the AI-powered glitch detection system is assessed through multiple evaluation criteria:

<u>Detection Accuracy</u> – The percentage of correctly identified glitches.

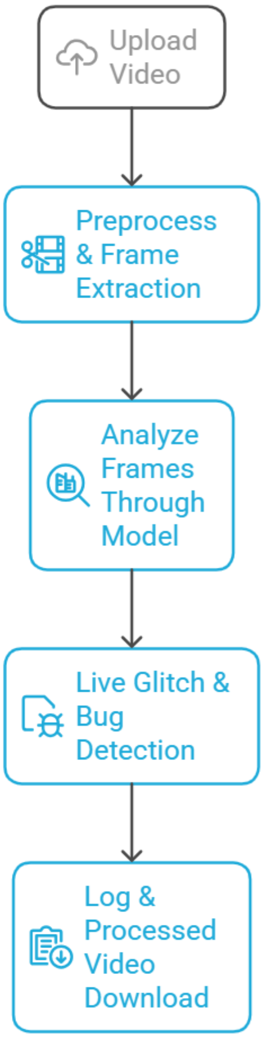
<u>False Positives & False Negatives</u> – To measure misclassification errors.

Processing Latency – Evaluating the real-time detection capability of the system.

Performance Evaluation & Feedback

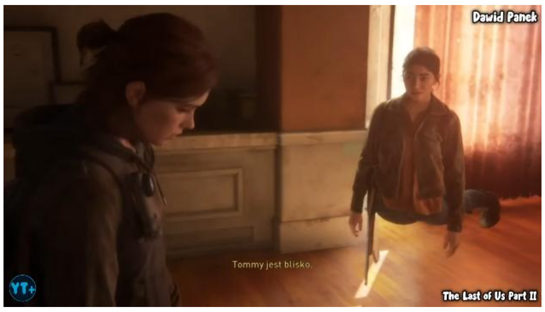
After each video processing session, the system generates a detailed performance report that highlights detected glitches with their corresponding timestamps, severity levels, and possible causes. Additionally, the report includes downloadable processed videos and logs for debugging. By offering a comprehensive analysis of video game glitches, BUGXAI enhances game testing through automated bug detection, ultimately helping developers optimize gameplay performance efficiently.

Working of the System

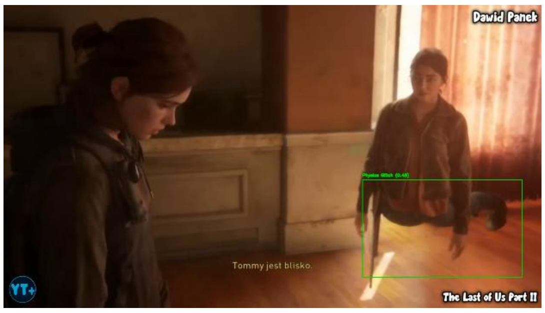


Upload Video
Preprocess & Frame Extraction
Analyze Frames Through Model
Live Glitch & Bug Detection
Log & Processed Video Download

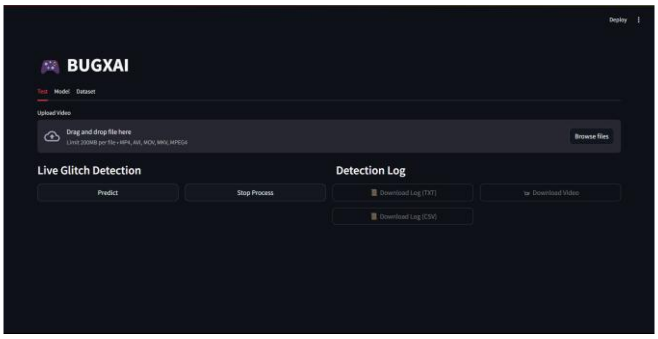
A Sample Frame from Input Video



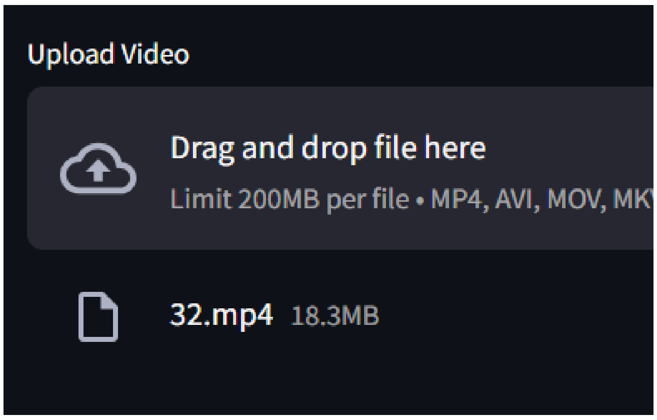
Output Frame of that Frame



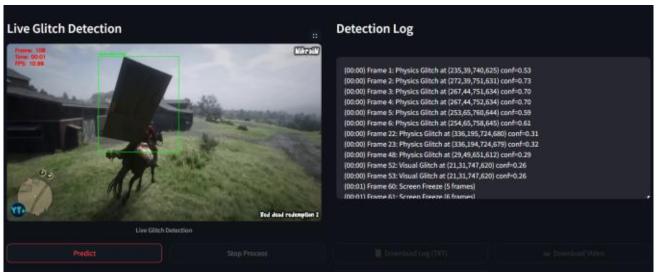
Result



The BUGXAI system's homepage serves as a central interface, enabling users to upload gameplay videos for automated bug detection. Upon uploading, the system preprocesses the video by extracting frames at a consistent rate, which are then analyzed using a deep learning model, specifically YOLOv8, to detect and classify glitches such as physics-based anomalies and visual inconsistencies. The real-time detection system identifies and logs these anomalies, allowing users to download a structured log file with timestamped glitch details alongside the processed video with annotated errors for further inspection. The backend efficiently handles computations, ensuring accurate detection with minimal latency, while the Streamlit-based user interface provides an intuitive experience for seamless interaction, visualization, and access to analytical insights. By automating the bug detection process, BUGXAI optimizes game testing workflows, reducing manual debugging efforts and enhancing overall game quality.

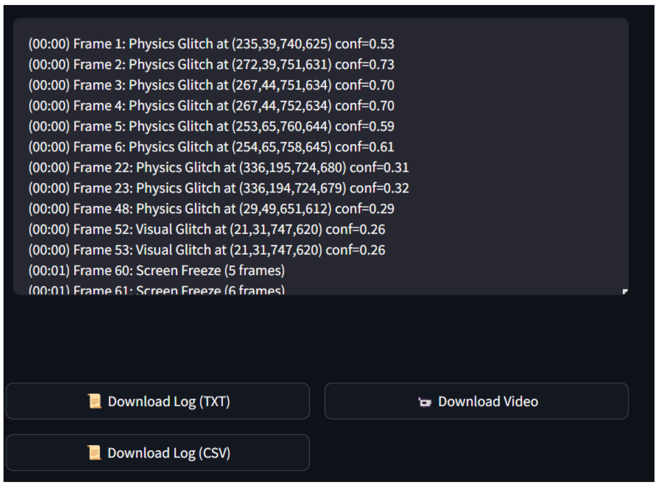


The uploaded file appears under the "Upload Video" section, indicating that the system has successfully received the video for processing. Once uploaded, the system will proceed with preprocessing, extracting frames, and running them through the YOLOv8 model to detect potential glitches. The user can then initiate the "Predict" function to analyze the video for bugs in real time, and upon completion, they will have the option to download the processed video and detection log for further review. This streamlined process ensures efficient and automated bug detection in gameplay footage.



```
(00:00) Frame 1: Physics Glitch at (235,39,740,625) conf=0.53
(00:00) Frame 2: Physics Glitch at (272,39,751,631) conf=0.73
(00:00) Frame 3: Physics Glitch at (267,44,751,634) conf=0.70
(00:00) Frame 4: Physics Glitch at (267,44,752,634) conf=0.70
(00:00) Frame 5: Physics Glitch at (253,65,760,644) conf=0.59
(00:00) Frame 6: Physics Glitch at (254,65,758,645) conf=0.61
(00:00) Frame 22: Physics Glitch at (336,195,724,680) conf=0.31
(00:00) Frame 23: Physics Glitch at (336,194,724,679) conf=0.32
(00:00) Frame 48: Physics Glitch at (29,49,651,612) conf=0.29
(00:00) Frame 52: Visual Glitch at (21,31,747,620) conf=0.26
(00:00) Frame 53: Visual Glitch at (21,31,747,620) conf=0.26
(00:01) Frame 60: Screen Freeze (5 frames)
(00:01) Frame 61: Screen Freeze (6 frames)
```

The glitch detection system processes the uploaded video frame by frame, identifying and classifying different types of glitches such as "Physics Glitch" and "Visual Glitch." The left side of the interface displays the current frame being analyzed, highlighting detected anomalies with bounding boxes and labels. Key metrics like the frame number, elapsed time, and frames per second (FPS) are displayed to provide real-time insights into the model's performance. The right side features a detailed detection log, listing each frame where a glitch was detected along with its precise coordinates and confidence score, enabling users to assess detection accuracy. The log also identifies recurring issues like "Screen Freeze" spanning multiple frames. Users can initiate or stop the detection process with the provided buttons and download the log in different formats for further analysis. This setup allows systematic tracking of graphical issues in videos, making it useful for quality assurance in gaming, video streaming, and software testing.



```
(00:00) Frame 1: Physics Glitch at (235,39,740,625) conf=0.53
(00:00) Frame 2: Physics Glitch at (272,39,751,631) conf=0.73
(00:00) Frame 3: Physics Glitch at (267,44,751,634) conf=0.70
(00:00) Frame 4: Physics Glitch at (267,44,752,634) conf=0.70
(00:00) Frame 5: Physics Glitch at (253,65,760,644) conf=0.59
(00:00) Frame 6: Physics Glitch at (254,65,758,645) conf=0.61
(00:00) Frame 22: Physics Glitch at (336,195,724,680) conf=0.31
(00:00) Frame 23: Physics Glitch at (336,194,724,679) conf=0.32
(00:00) Frame 48: Physics Glitch at (29,49,651,612) conf=0.29
(00:00) Frame 52: Visual Glitch at (21,31,747,620) conf=0.26
(00:00) Frame 53: Visual Glitch at (21,31,747,620) conf=0.26
(00:01) Frame 60: Screen Freeze (5 frames)
(00:01) Frame 61: Screen Freeze (6 frames)
```

The system offers a comprehensive glitch detection solution by analyzing individual frames for anomalies such as physics glitches, visual glitches, and screen freezes, logging their occurrences with frame coordinates and confidence scores. The right-side panel provides a real-time log of detected glitches, ensuring transparency and detailed tracking. Users can download the detection log in TXT or CSV formats, making it easier to analyze trends, validate detections, and integrate findings into other workflows. Additionally, the processed video can be downloaded, allowing for a visual review of glitches, which is particularly useful for debugging, performance optimization, and quality assurance in gaming and video applications.

Conclusion

This research introduces BUGXAI, an advanced AI-powered system for automated bug detection in video games. By leveraging computer vision, deep learning, and real-time anomaly detection, BUGXAI enhances the efficiency and accuracy of game testing, reducing manual debugging efforts for developers. The core contributions of BUGXAI include real-time video processing using YOLOv8 to detect glitches such as physics errors, frame drops, texture issues, and screen freezes; a comprehensive analysis framework that integrates semantic logging and anomaly classification to provide detailed debugging insights; and a user-friendly, Streamlit-based interface that enables game testers to upload videos, visualize detected glitches, and download structured logs and processed videos for further analysis. From a technical perspective, BUGXAI establishes a scalable and modular approach to automated game testing, setting a new benchmark for AI-assisted debugging solutions.

The research contributes to deep learning applications in video game quality assurance by offering an automated alternative to traditional bug-tracking systems, efficient dataset generation through the collection and labeling of diverse gaming footage, and a structured evaluation methodology that ensures reproducibility and accuracy in real-world scenarios. The implementation of BUGXAI demonstrates how AI can complement human game testers, improving efficiency without replacing human expertise, and as game environments become increasingly complex, AI-driven bug detection will play a pivotal role in enhancing gaming experiences by ensuring smoother and more immersive gameplay.

Future Scope

The BUGXAI system has several promising avenues for future development, aiming to enhance its capabilities and broaden its applications:

Reinforcement Learning for Automated Gameplay Testing

Implementing reinforcement learning (RL) to develop an AI agent that plays the game autonomously while our model continuously monitors the gameplay for bugs and glitches. This approach will enable the system to actively explore different game scenarios, stress-test mechanics, and improve bug detection accuracy in dynamic environments.

Expanding Dataset and Labeling

Increasing the dataset size by incorporating more labeled samples of different types of glitches, such as AI behavior inconsistencies, sound errors, and environment clipping. Training the model on a more diverse dataset to improve generalization across various game engines and genres.

Experimenting with Alternative AI Models

Testing other deep learning models beyond YOLOv8, such as transformer-based vision models, GANs for anomaly detection, or hybrid CNN-RNN architectures for better temporal analysis. Exploring self-supervised learning techniques to reduce dependency on manually labeled data.

Real-time Performance Optimization

Further refining the processing pipeline to achieve lower latency, making real-time bug detection even faster and more efficient. Implementing model quantization and hardware acceleration techniques (e.g., TensorRT, ONNX Runtime) to optimize inference speed without compromising accuracy.

Integration with Game Engines and Dev Tools

Developing plugins or APIs that integrate BUGXAI directly with game engines like Unity, Unreal Engine, and Godot. Enabling real-time debugging assistance for developers by highlighting detected glitches within the game development environment itself.

Multi-Modal Analysis for Enhanced Detection

Combining visual processing with other game telemetry data (e.g., physics engine logs, input actions) to detect subtle bugs that may not be visible in the video feed alone. Leveraging optical flow analysis and frame differencing to improve detection accuracy for frame drops and animation glitches.

Cloud-Based Bug Detection Service

Transforming BUGXAI into a cloud-based service where developers can upload gameplay footage for automated bug detection and receive detailed reports. Allowing large-scale testing of multiple game builds simultaneously.

References

Taesiri, M. R., Habibi, M., & Fazli, M. (2020). A video game testing method utilizing deep learning. *CSI Journal on Computer Science and Engineering,* *17*(2), 26–33. http://sina.sharif.edu/~fazli/papers/fazli-2020jcse.pdf

Varvaressos, S., Lavoie, K., Gaboury, S., & Hallé, S. (2017). Automated bug finding in video games: A case study for runtime monitoring. *Computers in Entertainment,* *15*(1), 1. https://dl.acm.org/doi/10.1145/2700529

Bergdahl, J., Gordillo, C., Tollmar, K., & Gisslén, L. (2021). Augmenting automated game testing with deep reinforcement learning. *arXiv preprint arXiv:2103.15819.* https://arxiv.org/abs/2103.15819

Azizi, E., & Zaman, L. (2023). Automatic bug detection in games using LSTM networks. *arXiv preprint arXiv:2312.08418.* https://arxiv.org/abs/2312.08418

Zarembo, Imants. (2019). ANALYSIS OF ARTIFICIAL INTELLIGENCE APPLICATIONS FOR AUTOMATED TESTING OF VIDEO GAMES. Environment. Technology. Resources. Proceedings of the International Scientific and Practical Conference. 2. 170. 10.17770/etr2019vol2.4158. https://www.researchgate.net/publication/333905048_ANALYSIS_OF_ARTIFICIAL_INTELLIGENCE_APPLICATIONS_FOR_AUTOMATED_TESTING_OF_VIDEO_GAMES

Senchenko, A. SUPERNOVA: Automating Test Selection and Defect Prevention in AAA Video Games Using Risk Based Testing and Machine Learning. 2022 IEEE Conference on Software Testing, Verification and Validation (ICST). https://www.academia.edu/112035047/SUPERNOVA_Automating_Test_Selection_and_Defect_Prevention_in_AAA_Video_Games_Using_Risk_Based_Testing_and_Machine_Learning