ISSN: 2664-9640

Automated Detection of 3D Game Visual Defects based on a Lightweight Vision-Language Model

Xiaohan Tu

School of Computer and Software, Nanjing University of Information Science & Technology, Nanjing, Jiangsu 210044, China

Abstract

Modern 3D games use complex rendering pipelines that make manual visual quality assurance expensive and difficult to scale. This study presents an automated screenshot-based defect detection method using a lightweight vision-language model. Qwen2.5-VL-7B is adapted with LoRA on a small domain dataset covering lighting, texture/material, and post-processing anomalies. Prompt constraints and Operational Design Domain cleaning are introduced to distinguish genuine defects from acceptable effects such as bloom, fog, and strong daylight. The adapted model is merged, converted to GGUF, quantized with Q4_K_M, and deployed offline through llama.cpp on an RTX 4060. On a 176-image evaluation set, the finalized configuration achieves 80.68% accuracy, 81.25% precision, and 70.27% recall, while reducing false positives to 12. The results indicate that few-shot adaptation and explicit task-boundary design can provide a low-cost, privacy-preserving pre-screening tool for 3D game quality assurance.

Keywords

Vision-language Model; Game Visual Defect Detection; LoRA Fine-tuning; Prompt Engineering; Local Deployment.

1. Introduction

1.1. Background and Significance

Game software combines the characteristics of a complex software system and a digital content product. Its quality depends not only on logical correctness but also on visual stability, continuity, and aesthetic consistency. With the widespread use of real-time global illumination, physically based materials, particle systems, dynamic resolution, and cross-platform rendering, visual problems such as missing textures, failed material loading, abnormal overexposure, broken shadows, and uncontrolled post-processing are more likely to arise under complex state combinations. Prior surveys show that game testing still relies heavily on manual play-testing and tester expertise, and insufficient automation remains a major obstacle to improving coverage and reducing repetitive work [1].

Traditional screenshot comparison and rule-based detection generally require stable reference frames, fixed cameras, and deterministic execution. Under legitimate visual variation caused by camera motion, smoke, dynamic lights, or artistic bloom, simple pixel differences can generate many false alarms. Vision-language models can jointly interpret image content and natural-language task boundaries, enabling a semantic judgment of whether an apparent anomaly is consistent with the scene. Compared with closed-set convolutional classifiers, a VLM can describe normal states, defect patterns, and exemption conditions through prompts and can produce structured explanations, making it suitable for assisted pre-screening in game QA.

Direct use of large general-purpose models nevertheless presents three difficulties. First, a general model lacks domain knowledge of game rendering and may mistake normal highlights, fog, and dark shadows for defects. Second, cloud APIs may require transmission of unreleased

ISSN: 2664-9640

game assets and therefore conflict with confidentiality requirements. Third, 7B-scale multimodal models impose substantial memory and storage costs. This study therefore performs LoRA adaptation on a small domain dataset and deploys a quantized model on a consumer GPU, forming an end-to-end workflow from data construction and model adaptation to local application.

1.2. Related Work

In game-testing research, the survey by Politowski et al. shows that manual play-testing remains the dominant practice and that automation is constrained by randomness, scenario coupling, and maintenance cost [1]. Paduraru and colleagues proposed hierarchical visual anomaly detection frameworks for commercial games, using a lightweight high-recall first stage followed by a high-precision verification stage and incorporating game state to reduce ambiguity caused by dynamic lighting and occlusion [2-3]. These approaches integrate well with game engines and continuous integration pipelines, but typically depend on large synthetic datasets, engine-side scripts, and structured state information. Zhou proposed GlitchAgent, which analyzes gameplay videos with multimodal large language models and uses local windows, temporal propagation, and semantic merging to describe and temporally localize anomalies [4], demonstrating the potential of MLLMs for open-ended game glitch understanding.

Table 1. Comparison of related studies

Study	Input/task	Core method	Strength	Limitation
Politowski et al. [1]	Game-testing survey	Survey of academic and grey literature	Systematic account of manual testing practice	No visual detection model
OwlEye [9]	Mobile GUI screenshots	CNN classification and Grad-CAM localization	Detects and localizes display defects	Targets UI rather than game-rendering semantics
WinCLIP [8]	Industrial anomaly images	Prompt ensembles and multi-scale CLIP	Strong zero-/few-shot performance	Not designed for game scenes or edge deployment
Paduraru et al. [2-3]	Game frames + state	Two-stage models and state fusion	Suitable for CI with controlled false alarms	Requires engine integration and large synthetic datasets
GlitchAgent [4]	Gameplay videos	MLLM local windows and temporal propagation	Describes and localizes temporal anomalies	High computation and deployment complexity
This study	Static game screenshots	Qwen2.5-VL LoRA + prompt/ODD + GGUF	Few-shot adaptation, explainable output, offline deployment	Small dataset; no cross-game or same-dataset benchmark validation; manual ODD cleaning; excludes temporal and geometric/physical defects

Note: The datasets and task settings differ across studies; the table therefore provides only a qualitative methodological comparison.

For general visual anomaly detection, OwlEye uses a deep visual model to identify and localize display defects in mobile application interfaces, demonstrating that screenshot-level visual understanding can support software testing [9]. WinCLIP improves zero-shot and few-shot anomaly classification and segmentation through ensembles of state words and prompt

ISSN: 2664-9640

templates, emphasizing that language can clarify the semantic boundary between normal and anomalous states [8]. Compared with industrial inspection and GUI defect detection, anomalies in game screenshots are more easily confused with legitimate artistic effects and therefore require stronger contextual understanding.

For multimodal instruction tuning, LLaVA connects a visual encoder to a language model and uses machine-generated image-text instruction data to develop general visual dialogue capabilities [5]. MobileVLM reduces multimodal inference cost through a lightweight vision-language architecture, visual-token compression, and an efficient projector [6]. MiniCPM, in contrast, shows from the perspective of scalable training and resource efficiency that models with fewer parameters can retain strong task capabilities [7]. These studies provide methodological support for the use of a 7B VLM, LoRA adaptation, and quantized deployment in this work.

Based on these gaps, this study addresses three research questions: whether a 7B vision-language model can be adapted with a limited number of game screenshots to distinguish genuine rendering defects from acceptable artistic effects; how prompt design and Operational Design Domain constraints change the precision-recall trade-off; and whether the adapted model can be quantized and deployed fully offline on an RTX 4060 while retaining practical detection performance.

2. Conceptual Foundations

2.1. Operational Design Domain

An Operational Design Domain (ODD) specifies the conditions under which a system is intended to operate and be evaluated. In this study, the ODD is restricted to single screenshots that contain an identifiable 3D game scene and sufficient visual evidence for judging rendering behavior. Samples outside these observable conditions are excluded rather than forced into the normal/defective binary decision.

Within the ODD, prompt rules define both defect evidence and exemptions for legitimate effects. Regular black bands, large material occlusions, and globally uncontrolled exposure are treated as high-confidence defect cues, whereas daylight glare, local bloom, smoke, thin fog, and intentional dark styling are not considered defects unless they make the main scene structurally unrecognizable.

2.2. Vision-language Model Decision Boundary

A vision-language model combines a visual encoder, a cross-modal projection module, and a language model. The visual encoder produces image tokens, the projector aligns them with the language-model embedding space, and the language model generates a task-conditioned response. This architecture allows screenshot content and natural-language constraints to be evaluated jointly.

In this work, language instructions function as part of the decision boundary rather than as a simple output template. They specify the included defect categories, normal-effect exemptions, and structured response fields. Detailed model selection, LoRA configuration, and quantized deployment procedures are therefore presented only in Sections 3.4, 3.5, and 5.6 to avoid repeating the related-work discussion.

3. Dataset Construction and Methodology

3.1. Overall Architecture

The system follows a three-layer architecture consisting of data engineering, model adaptation, and application deployment, as shown in figure 1. The data-engineering layer collects,

ISSN: 2664-9640

compresses, cleans, and annotates screenshots. The model-adaptation layer fine-tunes Qwen2.5-VL-7B with LoRA on AutoDL and performs adapter merging and GGUF conversion. The application layer starts a local llama.cpp service on an RTX 4060, which is called by a Gradio batch interface or a Tkinter real-time screenshot GUI. Expensive back-propagation is therefore concentrated in the cloud, while frequent and privacy-sensitive inference remains local.

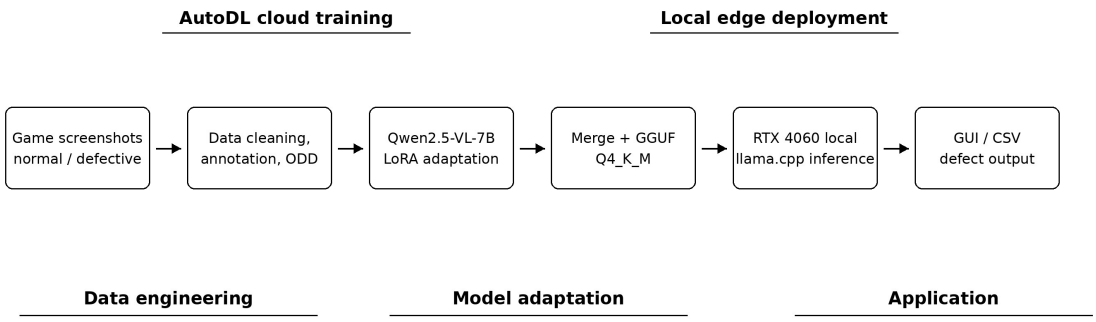


Figure 1. Overall architecture for cloud adaptation and local quantized deployment.

3.2. Data Collection, Organization, and Annotation

The dataset and evaluation follow a screenshot-level ODD. Included defects are limited to rendering anomalies visible in a single frame: lighting failures (such as uncontrolled exposure, light leaks, and broken shadows), texture/material failures (such as missing or pure-black materials, regular rectangular bands, and material misalignment), and post-processing failures (such as severe full-screen blur, color overflow, or rendering trails). Geometric, physical, temporal, and logic defects-including clipping, collisions, floating objects, flicker, animation desynchronization, and quest failures-are excluded.

The raw data consist of normal and defective game screenshots stored in the directories Normal_Images_Compressed and Defect_Images_Compressed. The corpus covers daylight glare, night scenes, smoke, bloom, local highlights, horizontal black bands, material anomalies, and severe overexposure. Data processing follows a semi-automatic pipeline of directory-label generation, stronger-model-assisted description, and human review. Directory membership provides the initial binary label, a vision-language model generates an initial defect description, and human reviewers correct mislabels, remove invalid screenshots, and standardize terminology.

ODD cleaning was implemented as a documented manual screening protocol rather than a fixed mean-luminance threshold, because intentionally dark night scenes are valid normal samples. A screenshot was excluded when (1) neither the main 3D subject nor the dominant scene structure could be identified at native resolution; (2) the image contained only a player frame, desktop tool, or other non-game content; (3) severe corruption prevented semantic inspection; or (4) it belonged to an excluded category such as clipping. The records show an approximately 196-image pre-cleaning pool, 176 retained images in the main evaluation set (74 defective and 102 normal), and 17 retained images in the independent set. Since the initial count was approximate and no versioned exclusion manifest was preserved, the exact number removed and category-wise removal counts cannot be reconstructed. The absence of a calibrated luminance threshold and complete cleaning log is therefore reported as a reproducibility limitation rather than being retrospectively assigned unsupported values.

ISSN: 2664-9640

Table 2. Dataset composition and purpose

Subset	Number of images	Purpose	Main content
Pre-cleaning pool	Approx. 196 (recorded)	ODD cleaning and task definition	Normal lighting, complex multi-light scenes, lighting/material/post-processing anomalies
Main evaluation set	176 (74 defective / 102 normal)	V1/V2.4/V2.6 comparison	Regular black bands, overexposure, dark scenes, smoke, normal highlights
Independent set	17	Post-finalization external evaluation	Unseen complex lighting and post-processing scenes

3.3. Multimodal Supervised Fine-tuning Format

Training data follow the multimodal ShareGPT format supported by LLaMA-Factory. Each sample contains an images field and a set of human/gpt messages. The user message introduces the screenshot with an <image> token and asks a defect-detection question, while the model response provides the defect status and a textual explanation. Compared with a class label alone, the image-instruction-response triplet provides supervision and constrains the model to generate a fixed output structure.

To reduce memorization of a single response pattern, training questions use multiple semantically equivalent formulations. Responses for normal samples emphasize that no definite defect is observed, while responses for defective samples describe the anomaly type, location, and evidence. All outputs are reviewed manually so that the boundary between normal highlights and abnormal overexposure remains consistent with the final deployment rules.

3.4. LoRA fine-tuning Configuration

Qwen2.5-VL-7B-Instruct is used as the backbone [11] and supervised fine-tuning is performed with LLaMA-Factory. Following the LoRA design of adapting query and value projections in attention layers [10], LoRA is applied to q_proj and v_proj with rank 8, scaling factor 16, and dropout 0.05. The per-device batch size is 1 and eight-step gradient accumulation produces a larger effective batch. The learning rate is 5×10^{-5} with cosine decay and a 5% warm-up, training runs for five epochs, and BF16 is used. Training is conducted on an AutoDL node equipped with an RTX 5090 with 32 GB of VRAM.

Table 3. Main LoRA fine-tuning hyperparameters

Configuration	Value	Description
Backbone	Qwen2.5-VL-7B-Instruct	7B multimodal instruction model
Fine-tuning method	LoRA SFT	Most backbone parameters frozen
LoRA target layers	q_proj, v_proj	Attention query and value projections
rank / alpha / dropout	8 / 16 / 0.05	Low-rank capacity and regularization
Batch size / gradient accumulation	1 / 8	Fits single-GPU memory
Learning rate / epochs	5×10^{-5} / 5	Cosine decay, warmup_ratio=0.05
Precision	BF16	Balances stability and memory use

3.5. Prompt Evolution and ODD Constraints

Three core configurations are evaluated. V1 is a zero-shot baseline that uses a general instruction to detect lighting, texture, and geometric problems. V2.4 strengthens the rule that suspected anomalies should be reported and therefore targets high recall after LoRA adaptation.

ISSN: 2664-9640

V2.6 explicitly excludes clipping, adds exemptions for smoke, thin fog, daylight glare, bright skies, local bloom, and legitimate artistic styles, and strengthens rules for regular black bands, large black blocks, and globally uncontrolled exposure that makes the main scene unrecognizable.

The final output is fixed to two sections, [ANALYSIS] and [RESULT]. [RESULT] contains Status, Type, and Reason. The structured output can be parsed into DEFECT_FOUND/NORMAL and the types Lighting, Texture, Material, Postprocess, and None, and can be recorded in CSV files for batch statistics.

Table 4. Prompt versions and operational preferences

Version	Operational preference	Key rules	Intended use
V1	General zero-shot	Broad detection of multiple visual problems	Backbone capability baseline
V2.4	High recall	Report even when evidence is uncertain	Machine screening + human review
V2.6	Balanced precision/stability	Normal-effect exemptions, documented ODD screening, strong black-band rules	Local assisted detection

4. Experimental Design

4.1. Hardware and Software Environment

The experiments use a cloud-edge environment. The cloud training node is an AutoDL RTX 5090 32 GB instance running Ubuntu 22.04 with Python 3.11, PyTorch CUDA 12.8, LLaMA-Factory, Transformers, and PEFT. The model, Conda environment, pip cache, and Hugging Face/ModelScope caches are all placed on the /root/autodl-tmp data disk. Local deployment uses Windows 11, an NVIDIA RTX 4060 with 8 GB of VRAM, Anaconda, llama.cpp, and a Python GUI.

Table 5. Experimental hardware and software environment

Layer	Hardware/system	Software/framework	Task
Cloud training	RTX 5090 32 GB, Ubuntu 22.04	PyTorch cu128, LLaMA-Factory, PEFT	LoRA fine-tuning, weight merging, GGUF conversion
Local inference	RTX 4060 8 GB, Windows 11	llama.cpp, Anaconda, Gradio, Tkinter	Batch inspection, real-time screenshots, CSV logging

4.2. Evaluation Metrics

Defective images are treated as the positive class and normal images as the negative class. TP denotes correctly detected defects, TN denotes correctly accepted normal images, FP denotes false alarms, and FN denotes missed defects. The main metrics are defined as follows:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN)$$
$$\text{Precision} = TP / (TP + FP)$$
$$\text{Recall} = TP / (TP + FN)$$
$$F1 = 2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall})$$

ISSN: 2664-9640

In game QA, precision represents the credibility of an alarm and recall represents the coverage of real defects. Excessive false positives create alarm fatigue and increase review cost, whereas excessive false negatives allow defects to reach later builds. No single metric is therefore treated as the sole objective. Instead, operating points are selected for different workflows: V2.4 is used for high-recall pre-screening, while V2.6 is used for local assisted detection when false-positive cost is high.

4.3. Experimental Procedure

First, V1, V2.4, and V2.6 perform image-by-image inference on the same main evaluation set, and the parsed Status field is compared with the human label. Second, confusion matrices and Accuracy, Precision, Recall, and F1 are calculated from the CSV results. Third, FP and FN samples are clustered into bad-case categories to examine the effects of dark regions, smoke, highlights, and black bands. Finally, the preliminary generalization of V2.6 is tested on 17 independent images, and GGUF inference, the batch interface, and the real-time screenshot interface are validated on the local RTX 4060.

5. Experimental Results and Analysis

5.1. Multi-stage Quantitative Results

Table 6 and figure 2 summarize the archived stage-level metrics. Because complete per-image records were not retained for every stage, V1 and the independent set are reported with explicit evidence-status notes and should not be interpreted as fully reproducible controlled ablations. V1 reaches 62.8% accuracy and 78.2% recall, but only 55.3% precision, indicating that the general model can identify anomaly cues but cannot reliably distinguish them from legitimate artistic effects. V2.4 increases recall to 89.9% and reduces the recorded FN count to 7, but precision drops to 48.1% and the recorded FP count increases to 67, reflecting its report-more high-recall screening strategy.

Through exemptions for normal effects and ODD cleaning, V2.6 reduces FP from 67 to 12 and raises accuracy and precision to 80.68% and 81.25%, respectively. Its recall is 70.27%, lower than V2.4, but its overall F1 rises to approximately 75.36%. The independent generalization set achieves 94.1% accuracy and 100.0% precision, but only 66.7% recall, showing that the model remains conservative on unseen samples and that the small-set result requires broader validation.

Table 6. Detection performance across stages

Stage	Accuracy/%	Precision/%	Recall/%	F1/%	FP	FN
V1 baseline†	62.8	55.3	78.2	64.8	55	Not retained
V2.4 screening	58.9	48.1	89.9	62.7	67	7
V2.6 finalized	80.68	81.25	70.27	75.36	12	22
Independent set‡	94.1	100.0	66.7	80.0	0	1 (derived)

Note: † The complete V1 per-image predictions were not retained. Its aggregate metrics and FP count are reproduced from the archived summary and should be treated as estimates; FN cannot be verified from the available files. ‡ For the 17-image independent set, FN=1 is derived from the reported 94.1% accuracy, 100.0% precision, 66.7% recall, and FP=0 rather than from a rerun. These rows are retained for transparency but do not provide fully reproducible confusion matrices. The V2.6 row is calculated from TP=52, TN=90, FP=12, and FN=22.

ISSN: 2664-9640

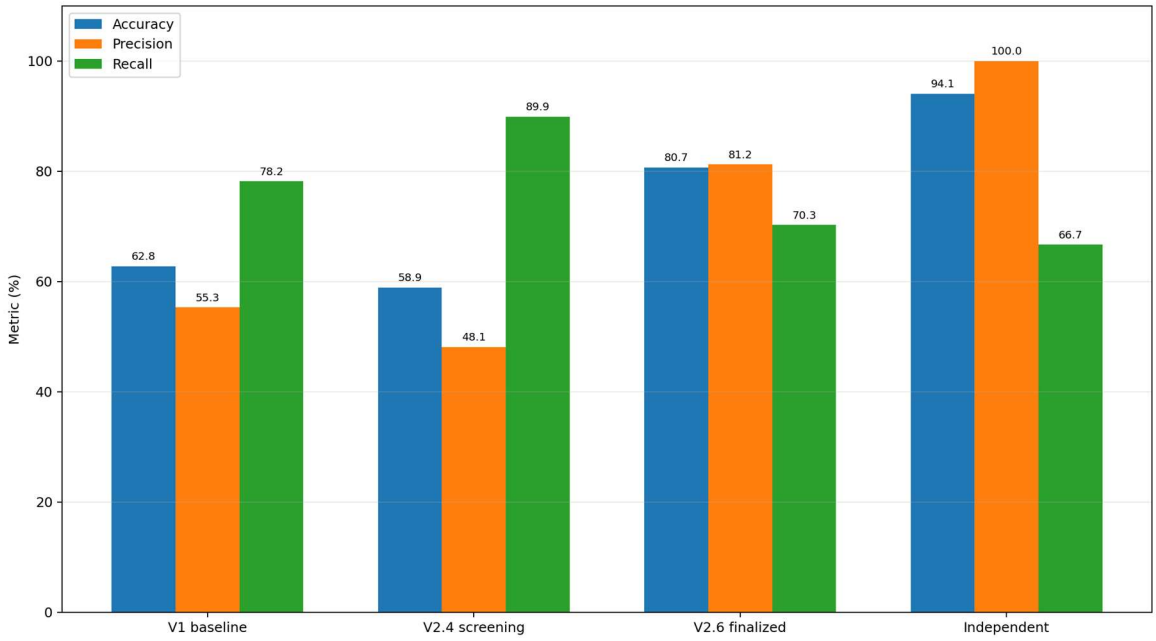


Figure 2. Accuracy, precision, and recall across experimental stages.

5.2. False-positive Control

In automated QA, false positives directly determine the human-review burden. Figure 3 shows that V1 produces 55 FP cases without domain constraints. V2.4 increases FP to 67 because normal highlights, smoke, and local blur are treated as possible anomalies. After V2.6 encodes normal effects as exemptions and removes extremely dark low-information screenshots outside the ODD, FP falls to 12. No FP occurs in the independent set. These results show that increasing model sensitivity alone does not improve system usability; defect detection must constrain false-positive cost according to the deployment workflow.

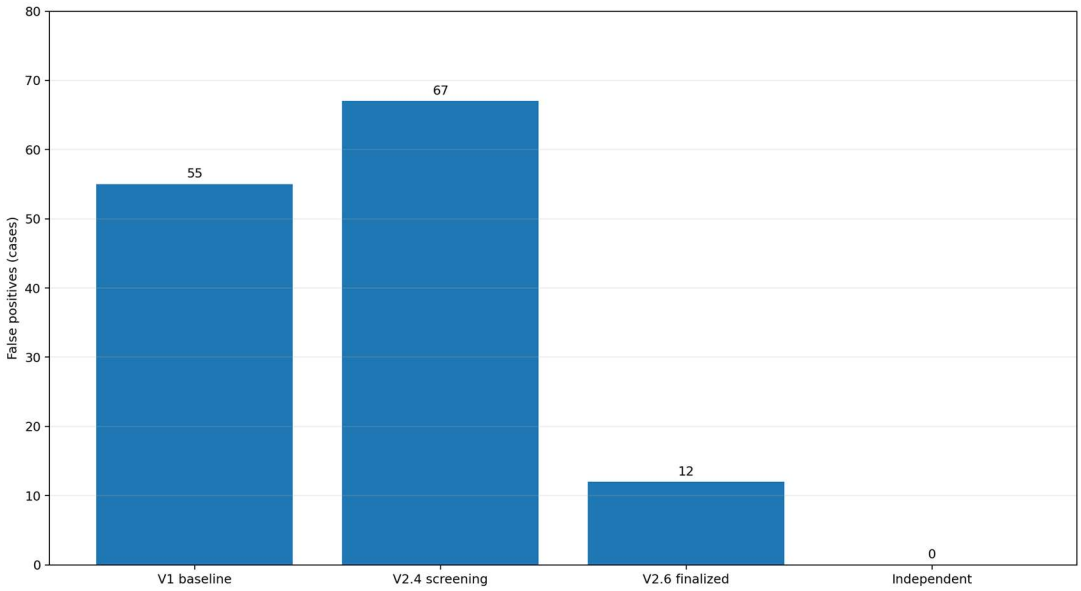


Figure 3. Number of false positives across experimental stages.

5.3. V2.6 Confusion-matrix Analysis

The V2.6 main evaluation set contains 74 defective and 102 normal images. The model obtains TP=52, FN=22, FP=12, and TN=90, as shown in figure 4. The high TN indicates that most normal scenes are correctly accepted, especially daylight glare, smoke, and local bloom. TP cases are

ISSN: 2664-9640

concentrated in multiple regular black horizontal bands, large material occlusions, and severe exposure failures that make the subject unrecognizable.

The 22 FN cases mainly occur in low-contrast anomalies, extremely dark scenes, and ambiguous local highlights. Four-bit quantization may suppress some shadow detail, while the conservative V2.6 instruction that insufficient evidence should be classified as normal deliberately sacrifices recall for weak defects. This outcome is consistent with the design goal of assisted pre-screening rather than complete replacement of human testers.

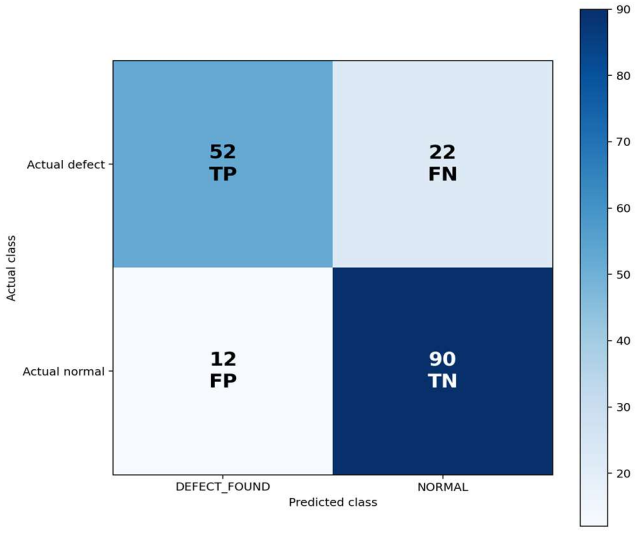
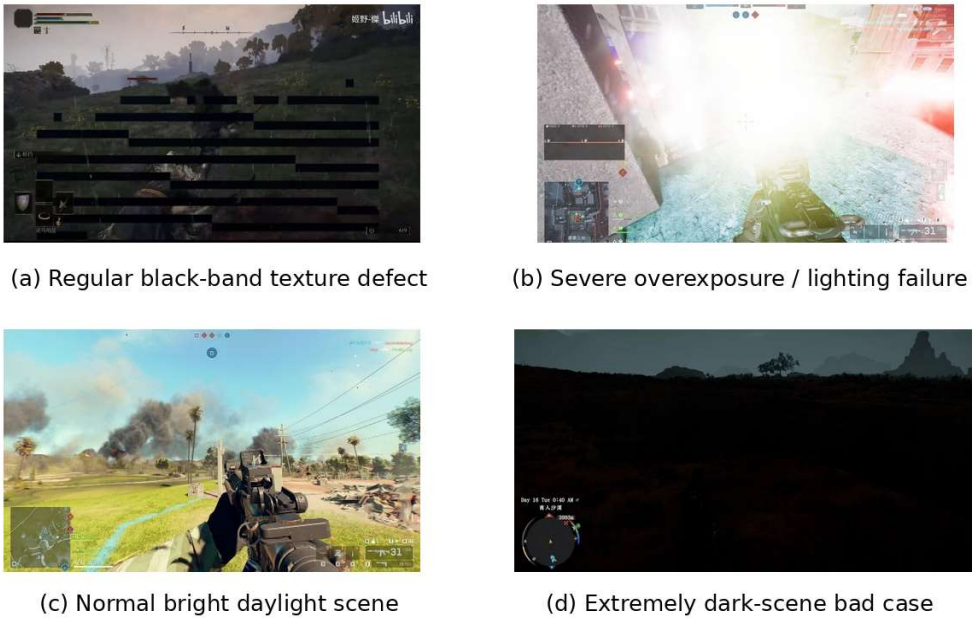


Figure 4. Confusion matrix of the finalized V2.6 model.

5.4. Representative Cases and Bad Cases



(a) Regular black-band texture defect

(b) Severe overexposure / lighting failure

(c) Normal bright daylight scene

(d) Extremely dark-scene bad case

Figure 5. Representative defects, normal samples, and bad cases. Images are from the self-constructed experimental dataset.

Figure 5 presents representative samples. Regular black bands are nearly pure black, have straight rectangular boundaries, and cover the 3D subject; the model can generally classify them consistently as Texture or Material defects. Large-scale overexposure is classified as

ISSN: 2664-9640

Lighting or Postprocess when characters, terrain, and scene structure become largely unrecognizable. In contrast, ordinary daylight highlights, explosion smoke, and local reflections should be classified as NORMAL.

Extremely dark scenes are the main bad-case category. A normal night environment may be mistaken for missing textures because of block-like dark regions, while a genuine dark-region defect may be missed because there is insufficient detail. Future work can add luminance-distribution checks, shadow enhancement, and low-visibility tags before the model and introduce “suspected defect” as a third state to reduce errors caused by an overly rigid binary boundary.

5.5. Independent Generalization Evaluation

The 17 independent test images were not used in fine-tuning or prompt iteration. V2.6 achieves 94.1% accuracy, 100.0% precision, and 66.7% recall on this set. Together with FP=0, these rounded aggregate values imply TP=2, TN=14, and FN=1; because the per-image prediction log was not retained, the counts are reported as derived rather than rerun results. The absence of false positives indicates that the exemptions for normal effects transfer to some unseen lighting conditions, while the lower recall shows that the conservative threshold still misses weak anomalies. Because the set is small and the counts are reconstructed from summary metrics, the result should be interpreted only as a preliminary trend and cannot replace large-scale cross-game, cross-engine, and cross-platform evaluation.

5.6. Edge-deployment Results

After the LoRA adapter is merged with the backbone, the model is converted to GGUF. The Q4_K_M main model is approximately 4.4 GB, the F16 mmproj is approximately 1.3 GB, the deployment package totals approximately 5.7 GB, and the adapter archive is approximately 214 MB. The RTX 4060 loads the main model and mmproj simultaneously through llama.cpp with an 8192-token context. Screenshots are compressed to a longest side of about 768 pixels and a size of tens of kilobytes before being sent to the local endpoint, reducing visual-token and transfer cost and avoiding context overflow.

Table 7. Model-training and deployment artifacts

Artifact	Size	Purpose
LoRA adapter	Approx. 214 MB	Training archive, continued training, reproducibility
Q4_K_M main-model GGUF	Approx. 4.4 GB	Main language and multimodal inference weights
F16 mmproj GGUF	Approx. 1.3 GB	Visual encoding projection module
Local deployment package	Approx. 5.7 GB	Offline RTX 4060 deployment

6. Implementation of the Interactive Detection System

6.1. Local model Service

llama-server.exe loads the Q4_K_M main model with -m, loads the visual projector with --mmproj, and registers the model name qwen25vl-game-defect with --alias. The service listens on 127.0.0.1:8080 and supports OpenAI Chat Completions requests. Before use, the application can query /health and /v1/models. The context size is set to 8192 to accommodate the visual tokens and prompt. If the consumer GPU lacks sufficient VRAM, the number of GPU-offloaded layers can be reduced so that part of the computation falls back to the CPU.

ISSN: 2664-9640

6.2. Batch Image Inspection

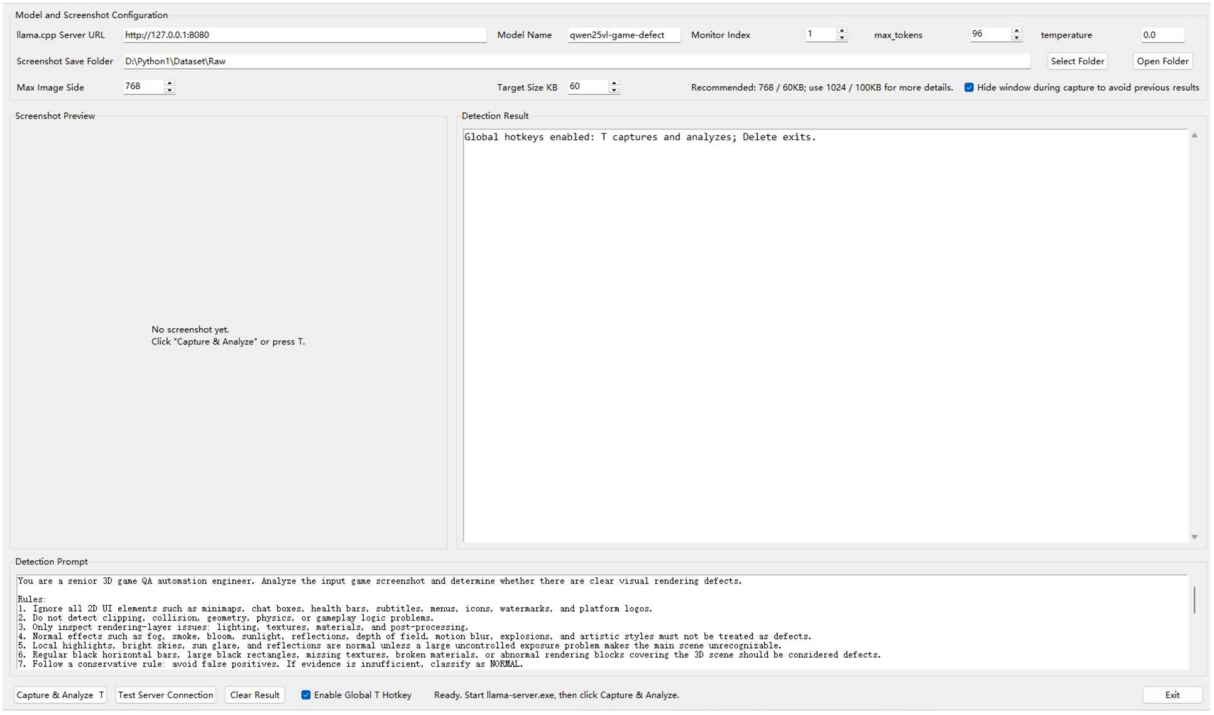
The Gradio version supports multiple-image upload, image compression, model-name configuration, and prompt configuration. It calls the local service image by image and writes the human label, original filename, image sizes before and after compression, predicted status, defect type, reason, elapsed time, and raw output to CSV. Batch mode is suitable for fixed test sets, ablation experiments, and bad-case analysis, and TP, TN, FP, and FN can be calculated directly from the CSV file.

6.3. Real-time Screenshot Inspection

The Tkinter GUI supports screenshot capture through a button or the global T hotkey, while Delete exits the program. To reduce visual tokens, screenshots are resized in memory and compressed to JPEG files of tens of kilobytes. The detection window is temporarily hidden during capture so that previous output is not captured and repeated by the model. Hotkey debouncing and a busy-state lock prevent multiple inference threads from being started by a held T key. After inference, only the first Status, Type, and Reason group is parsed to prevent repeated model output from flooding the interface.

6.4. Structured Output and Engineering Value

The final response contains [ANALYSIS] and [RESULT]. The former uses one or two sentences to describe the main lighting, material, texture, or post-processing behavior, while the latter outputs a fixed status, type, and reason. This design supports both interpretability and machine parsing: QA engineers can inspect the model rationale, while software can write the category to CSV or trigger a downstream ticket. Because inference is entirely local, unreleased game assets do not need to be uploaded to the public Internet, making the system suitable for development environments with strict confidentiality requirements.



ISSN: 2664-9640

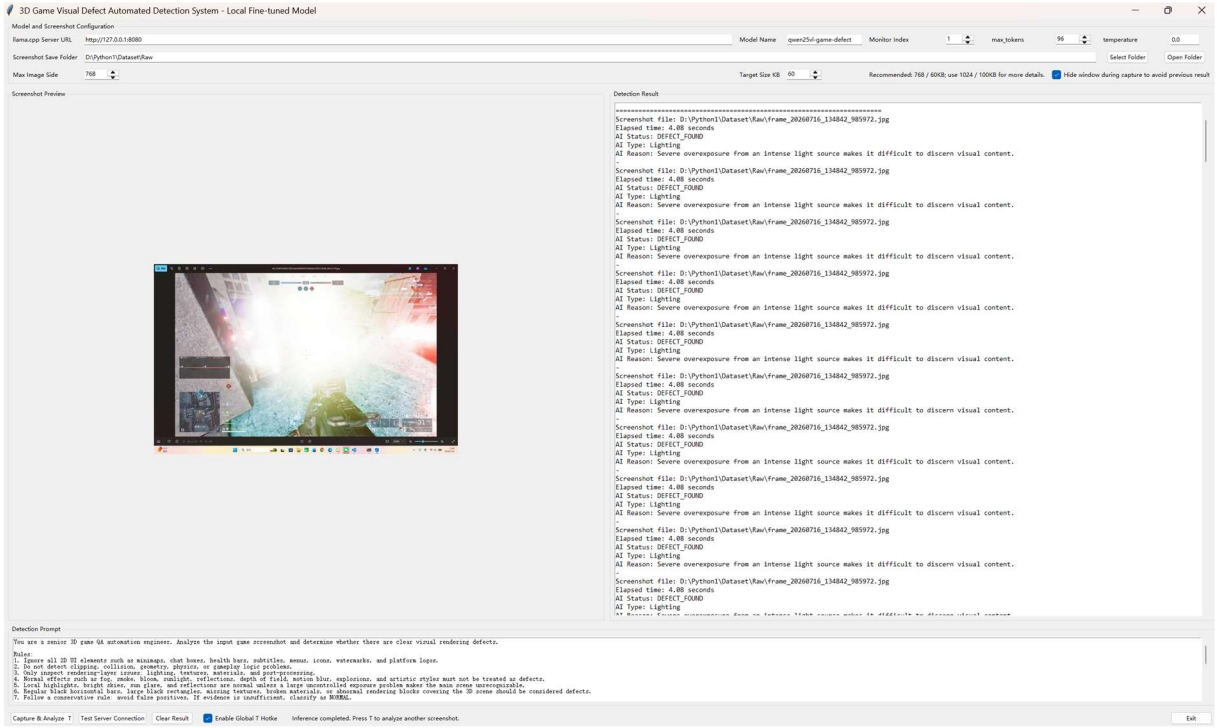


Figure 6. Real-time screenshot inspection: idle state (top) and severe-overexposure result (bottom).

7. Discussion and Threats to Validity

7.1. Method Effectiveness and Applicable Scenarios

The experimental results show that the value of a VLM in game visual-defect detection comes not only from visual encoding but also from the ability of natural language to define task boundaries. V2.4 and V2.6 use the same adapted model but produce different operating points, showing that prompts can substantially change the alarm strategy. For large-scale pre-release screenshot screening, V2.4 can be selected to reduce missed defects followed by human review. For continuously running local tools, V2.6 can reduce false alarms and alarm fatigue.

Compared with hierarchical detection frameworks that require engine state and large synthetic datasets [2-3], the proposed approach has a lower deployment barrier and requires only screenshots and natural-language labels. Compared with video-level GlitchAgent [4], it has lower computational cost and can run on a consumer GPU. The lower barrier also means that the system lacks game state, reference frames, and temporal continuity, so it is more appropriate as a pre-screening tool for rendering problems.

7.2. Limitations

The dataset and generalization evidence are limited. The main evaluation set contains only 176 images and the independent set contains only 17 images, which cannot adequately cover different games, engines, artistic styles, and display platforms. No cross-game validation was conducted, and the approximate pre-cleaning count and missing exclusion manifest prevent exact reconstruction of the cleaning process.

The task uses only single screenshots. Flicker, texture pop-in, animation desynchronization, short-lived occlusion, and physical anomalies require consecutive frames and temporal reasoning and are difficult to detect reliably from one frame.

Prompts and model weights are coupled. Exemptions for normal effects reduce FP but may interpret weak overexposure as normal and thereby increase FN.

ISSN: 2664-9640

Quantization may affect dark-region and fine-grained visual information. Q4_K_M enables local deployment but may introduce some accuracy loss compared with BF16/FP16 inference.

The system does not detect clipping, collisions, floating objects, or logic defects and should not be interpreted as a general detector for all game bugs.

The study lacks a direct benchmark comparison under the same dataset, input modality, and hardware. Table 1 is therefore a qualitative methodological comparison and does not establish superiority over prior approaches.

Some development-stage prediction logs were not retained. In particular, the V1 FN count cannot be audited and the independent-set confusion counts are derived from rounded aggregate metrics, limiting the reproducibility of cross-stage comparisons.

7.3. Threats to Validity

For internal validity, prompt changes, data cleaning, and fine-tuning differ across versions, making it difficult to isolate the contribution of each factor. Future work should fix model weights, conduct single-variable prompt ablations, and repeat inference on the same test set. For external validity, the data come from a limited set of game scenes and cross-game generalization has not been sufficiently validated. For construct validity, binary labels cannot represent defect severity or uncertainty, suggesting the addition of a suspected-defect state and calibrated confidence. For conclusion validity, the independent set is small and the confidence interval around 94.1% accuracy is wide, so the result should not be interpreted as universal industrial performance.

8. Conclusion and Future Work

This study develops a lightweight visual-defect detection method and local application prototype based on Qwen2.5-VL-7B for lighting, texture/material, and post-processing anomalies in 3D game screenshots. LoRA is used for few-shot domain adaptation, and multi-stage experiments with V1, V2.4, and V2.6 demonstrate the substantial effect of prompt strategy on the precision-recall trade-off. On the 176-image main evaluation set, V2.6 achieves 80.68% accuracy, 81.25% precision, and 70.27% recall with 12 false positives. On the 17-image independent set, it achieves 94.1% accuracy and 100.0% precision.

From an engineering perspective, the study completes an end-to-end workflow from data organization and LoRA fine-tuning through adapter merging, GGUF conversion, Q4_K_M quantization, and local inference on an RTX 4060, and implements both batch inspection and a real-time screenshot GUI. The results show that cloud training combined with local inference can control cost while satisfying privacy requirements for game assets.

Future work will proceed in four directions. First, cross-game, cross-engine, and cross-platform data will be expanded with fine-grained defect types and severity annotations. Second, video windows, optical flow, or Video Transformers will be introduced to detect flicker, pop-in, and animation anomalies. Third, reference frames, game state, or scene graphs will be integrated to improve understanding of clipping, floating objects, and physical problems. Fourth, mixed-precision quantization, visual-projector quantization, and confidence calibration will be studied to improve weak-defect recall while maintaining deployability on an RTX 4060.

References

[1] Politowski, C., Petrillo, F., & Guéhéneuc, Y. G. (2021). A survey of video game testing. arXiv preprint arXiv:2103.06431.

[2] Păduraru, C. I., Păduraru, M. G., & Stefanescu, A. (2025). Hierarchical deep learning framework for continuous, state-aware visual glitch detection in games. In *Evaluation and Assessment in Software Engineering (EASE 2025)* (Istanbul, Türkiye). https://doi.org/10.1145/3756681.3757010

ISSN: 2664-9640

[3] Paduraru, C. (2026). A state-aware, hierarchical deep learning framework for automated visual glitch detection in games. *Engineering Applications of Artificial Intelligence*, 166, 113497. https://doi.org/10.1016/j.engappai.2025.113497

[4] Zhou, T. (2025). *GlitchAgent: Detecting Video Game Glitches from Gameplay Videos* [Master's thesis]. Virginia Polytechnic Institute and State University, Blacksburg.

[5] Liu, H., Li, C., Wu, Q., & Lee, Y. J. (2023). Visual instruction tuning. In *Advances in Neural Information Processing Systems* (Vol. 36).

[6] Chu, X., Qiao, L., Lin, X., et al. (2023). MobileVLM: A fast, strong and open vision language assistant for mobile devices. arXiv preprint arXiv:2312.16886.

[7] Hu, S., Tu, Y., Han, X., et al. (2024). MiniCPM: Unveiling the potential of small language models with scalable training strategies. arXiv preprint arXiv:2404.06395.

[8] Jeong, J., Zou, Y., Kim, T., et al. (2023). WinCLIP: Zero-/few-shot anomaly classification and segmentation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 19606–19616).

[9] Liu, Z., Chen, C., Wang, J., et al. (2020). Owl eyes: Spotting UI display issues via visual understanding. In *35th IEEE/ACM International Conference on Automated Software Engineering* (Virtual Event, Australia). https://doi.org/10.1145/3324884.3416547

[10] Hu, E. J., Shen, Y., Wallis, P., et al. (2022). LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

[11] Qwen Team. (2025). Qwen2.5-VL technical report. arXiv preprint arXiv:2502.13923.

ISSN: 2664-9640

Appendix A. Key deployment parameters and output format

Table A-1. Key local deployment parameters

Item	Parameter
llama.cpp service	host=127.0.0.1, port=8080, context=8192, model alias=qwen25vl-game-defect
Main model	qwen25vl-game-defect-Q4_K_M.gguf (approx. 4.4 GB)
Visual projector	mmproj-qwen25vl-game-defect-f16.gguf (approx. 1.3 GB)
Real-time preprocessing	Longest side approx. 768 px, target approx. 60 KB, JPEG
Generation parameters	temperature=0.0, max_tokens≈96, repeat_penalty≈1.2
Structured result	[RESULT] Status / Type / Reason

In practical use, model predictions should be treated as automated pre-screening results. Samples classified as DEFECT_FOUND should be reviewed by QA engineers using the original screenshot, adjacent frames, and game state. Dark or low-contrast scenes may be enhanced or routed to a suspected-defect queue.