

**Leveraging Foundation Models for Video Game Quality Assurance**

by

Mohammad Reza Taesiri

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering  
University of Alberta

© Mohammad Reza Taesiri, 2024

# Abstract

The video game industry has become a powerhouse in the global entertainment economy. Creating engaging, high-quality games demands intricate development processes and significant resources. As projects grow in complexity and scale, developers often grapple with demanding schedules, tight deadlines, and the risk of burnout. These pressures highlight the need for more efficient development strategies, with quality assurance (QA) emerging as a critical area for optimization.

Artificial Intelligence (AI) has the potential to address these challenges by enhancing the game QA processes in large gaming companies. Specifically, foundation models—large pre-trained AI models—offer promising applications to improve these processes. Exploring novel uses of these advanced AI models could reveal their potential and limitations in optimizing game development workflows, potentially alleviating some of the industry’s pressing issues and facilitating the creation of high-quality, engaging games.

In this thesis, my goal is to improve video game testing processes by leveraging foundation models to ensure the final product reaches a desirable quality. I explore new opportunities that foundation models bring to game testing, from searching for instances of game bugs within video repositories to assisting human testers in catching bugs, through three studies:

First, I investigate the utility of image-text foundation models in retrieving gameplay videos. In this study, I create a video search engine designed to help developers efficiently search video repositories for examples of video game bugs using textual descriptions. For example, developers can find all instances of a bug by using a tex-

tual description of the bug, such as *a horse flying in the air*. This study lays the groundwork for AI-based game QA processes, with results demonstrating significant potential.

Next, I introduce **GlitchBench**, a benchmarking dataset of video game glitches and anomalies designed to assess state-of-the-art large multimodal models, such as GPT-4V, in detecting and understanding game bugs. This extensive dataset includes a wide range of images depicting various glitches, sourced from both online platforms and synthetic sets created within the Unity game engine. **GlitchBench** includes both common and rare glitches encountered in the video game quality assurance process. The findings from this study highlight both the promise and limitations of existing models, particularly in unusual and rare cases.

Lastly, I introduce **VIDEOGAMEBUNNY**, a large multimodal model specifically trained for video game content, accompanied by a dataset of 389,565 image-instruction pairs. My analysis demonstrates that **VIDEOGAMEBUNNY** outperforms much larger models in video game understanding tasks while using  $4.2\times$  fewer parameters. This result underscores the effectiveness and promise of using a high-quality dataset to improve models' understanding of video games, thus making them more effective in the game QA process.

Future work should focus on enhancing the generalization and robustness of AI models in the gaming context, particularly through better integration of vision and language components. This integration could be achieved using either early or late fusion methods. For late fusion methods, where two pre-trained models are connected, better alignment between these components can be achieved through improved training data and strategies. Alternatively, early fusion techniques, which involve training both components simultaneously to enhance their integration, can overcome many issues that existing models have.

# Preface

The research work presented in this thesis has been conducted in the Analytics of Software, GAMES, And Repository Data (ASGAARD) lab led by Dr. Cor-Paul Bezemer. This thesis is an original work by Mohammad Reza Taesiri.

Chapter 3 was published as: Taesiri, Mohammad Reza, et al. “Searching bug instances in gameplay video repositories.” in IEEE Transactions on Games (2024) [177]. This is a follow-up work, in collaboration with our industry partner, which was originally published as: Taesiri, Mohammad Reza, Finlay Macklon, and Cor-Paul Bezemer. “CLIP meets GamePhysics: towards bug identification in gameplay videos using zero-shot transfer learning.” [176] in 2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR) (2022). I was responsible for developing the ideas, collecting the data, designing the benchmark, evaluating the various CLIP models, and manuscript composition. F. Macklon was involved in manuscript composition in the original study. Dr. Sarra Habchi was involved in concept formation for the second study. Dr. Bezemer was the supervisory author for both studies and was involved in concept formation and manuscript composition. Part of this work was done in collaboration with Ubisoft.

Chapter 4 has been published as: Taesiri, Mohammad Reza, et al. “GlitchBench: Can large multimodal models detect video game glitches?” [174] in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2024. I was responsible for developing the ideas, collecting and processing the video game glitches, designing the benchmark, evaluating various state-of-the-art large multimodal models, and manuscript composition. T. Feng was responsible for creating

some synthetic video game glitches inside the Unity game engine. Dr. Anh Nguyen and Dr. Bezemer were both the supervisory authors and were involved in concept formation and manuscript composition.

Chapter 5, “`VIDEOGAMEBUNNY`: Towards Vision Assistants for Video Games,” [173] has been published in the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2025. I was responsible for developing the ideas, collecting the data, training `VIDEOGAMEBUNNY`, designing the benchmark, evaluating various large multimodal models, and manuscript composition. Dr. Bezemer was the supervisory author and was involved in concept formation and manuscript composition.

# Table of Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Foundation models – A new paradigm . . . . .	2
1.2	Foundation models for video game quality assurance . . . . .	3
1.3	Thesis objectives . . . . .	5
1.4	Thesis outline . . . . .	8
<b>2</b>	<b>Background</b>	<b>10</b>
2.1	Video game glitches . . . . .	10
2.1.1	Notable examples of glitches . . . . .	10
2.1.2	The challenge of glitch detection in game development . . . . .	11
2.2	Vision-language models . . . . .	13
<b>3</b>	<b>Searching bug instances in gameplay video repositories</b>	<b>18</b>
3.1	Introduction . . . . .	19
3.2	Motivation and background . . . . .	22
3.2.1	Video game (physics) bugs . . . . .	22
3.2.2	Challenges in mining gameplay videos . . . . .	23
3.2.3	Contrastive learning and zero-shot transfer . . . . .	23
3.2.4	The Contrastive Language-Image Pre-Training (CLIP) model . . . . .	24
3.3	Related work . . . . .	25
3.4	Our approach . . . . .	27
3.4.1	Encoding video frames and the text query . . . . .	27
3.4.2	Calculating the similarity of embeddings . . . . .	27
3.4.3	Aggregating frame scores per video . . . . .	28
3.5	Preparing the GamePhysics dataset . . . . .	29
3.5.1	Collecting the GamePhysics dataset . . . . .	29
3.5.2	Pre-processing the videos . . . . .	31
3.6	Preliminary evaluation setup . . . . .	31
3.6.1	Experiment overview . . . . .	31

3.6.2	Selecting CLIP architectures . . . . .	31
3.6.3	Selecting video games . . . . .	32
3.6.4	Query formulation . . . . .	33
3.6.5	Preliminary experiment 1: simple queries . . . . .	33
3.6.6	Preliminary experiment 2: compound queries . . . . .	33
3.6.7	Evaluating the experiments . . . . .	34
3.6.8	Results for preliminary experiments . . . . .	34
3.7	In-depth evaluation . . . . .	36
3.7.1	Experiment setup . . . . .	37
3.7.2	Dataset and labeling videos . . . . .	37
3.7.3	Models . . . . .	38
3.7.4	Experimental results . . . . .	41
3.7.5	GPU memory usage and search speed analysis . . . . .	43
3.8	Discussion and limitations . . . . .	45
3.9	Threats to validity . . . . .	49
3.9.1	Threats to internal validity . . . . .	50
3.9.2	Threats to external validity . . . . .	50
3.10	Conclusion . . . . .	51
<b>4</b>	<b>GlitchBench: Can large multimodal models detect video game glitches?</b>	<b>53</b>
4.1	Introduction . . . . .	54
4.2	Related work . . . . .	56
4.2.1	Multimodal, image-text datasets . . . . .	56
4.2.2	Vision-language stress testing . . . . .	58
4.2.3	Empirical analysis of recent LMMs . . . . .	59
4.3	GlitchBench . . . . .	60
4.3.1	Constructing the dataset . . . . .	60
4.3.2	Labeling the dataset . . . . .	62
4.3.3	Categorizing the glitch types in the images . . . . .	62
4.4	Experiments . . . . .	64
4.4.1	Experimental setup . . . . .	64
4.4.2	Quantitative results . . . . .	66
4.4.3	Qualitative observations and analysis . . . . .	68
4.5	Discussion and limitation . . . . .	73
4.6	Conclusion . . . . .	73

<b>5</b>	<b>VIDEOGAMEBUNNY: Towards vision assistants for video games</b>	<b>75</b>
5.1	Introduction . . . . .	76
5.2	Background and related work . . . . .	78
5.2.1	Large multimodal models . . . . .	78
5.2.2	Instruction following data . . . . .	79
5.2.3	LLMs and LMMs in video games . . . . .	79
5.2.4	Empirical analysis of large multimodal models . . . . .	80
5.3	VIDEOGAMEBUNNY model architecture . . . . .	80
5.4	Instruction-following data for video game content . . . . .	81
5.4.1	Video game images . . . . .	81
5.4.2	Generating instructions . . . . .	83
5.4.3	Evaluation dataset . . . . .	87
5.5	Experiments . . . . .	87
5.6	Results . . . . .	90
5.7	Discussion . . . . .	93
5.8	Conclusion . . . . .	97
<b>6</b>	<b>Discussions, conclusion and future work</b>	<b>98</b>
6.1	Discussion . . . . .	98
6.1.1	Threats to validity . . . . .	98
6.2	Conclusion . . . . .	101
6.3	Future work . . . . .	103
6.3.1	Model improvements and architectural changes . . . . .	103
6.3.2	Applications . . . . .	104
	<b>Bibliography</b>	<b>106</b>
	<b>Appendix A: Appendix GLITCHBENCH For Can large multimodal models detect video game glitches?</b>	<b>129</b>
A.1	Implementation details . . . . .	129
A.1.1	Details about model inference . . . . .	129
A.1.2	Details about the judge . . . . .	129
A.2	Additional results . . . . .	131
A.2.1	Breakdown of performance by various glitch types . . . . .	131
A.2.2	Using multiple frames . . . . .	131
A.3	Glossary of model failures . . . . .	132
A.3.1	Failures related to reasoning about the content of the image . . . . .	133
A.3.2	Failures related to facial features . . . . .	134

A.3.3	Failures related to unnatural body positions . . . . .	137
A.3.4	Failures related to multimodal hallucination . . . . .	141
A.3.5	Failures related to the judge accepting the wrong answer . . .	148
A.3.6	Failures related to clipping issues . . . . .	153
A.3.7	Failures related to unusual circumstances . . . . .	159
A.4	Sample glitches that are hard for humans . . . . .	162
A.5	Synthetic sample generated with Unity . . . . .	166

**Appendix B: Appendix For VIDEOGAMEBUNNY: Towards visual assistants for video games** **168**

B.1	Additional details . . . . .	168
B.1.1	Prompts used to generated datasets . . . . .	168
B.2	Additional results . . . . .	171
B.3	Qualitative samples . . . . .	173
B.4	Sample image-to-JSON conversion . . . . .	179

# List of Tables

3.1	Selected games for the preliminary evaluation of our approach. All selected games are open-world. . . . .	32
3.2	Average Top- $k$ accuracy (%) per game for simple queries (Preliminary Experiment 1). . . . .	34
3.3	Average Top- $k$ accuracy (%) per game for compound queries (Preliminary Experiment 2). . . . .	34
3.4	Average Top- $k$ accuracy (%) per query for simple queries (Preliminary Experiment 1). $N$ is the number of games searched. . . . .	35
3.5	Sample queries . . . . .	38
3.6	The performance of the studied models on our labeled dataset of 220 videos – The bold numbers highlight the best-performing model. (%)	44
3.7	GPU memory usage of the studied models. . . . .	45
4.1	Categorization of video game glitches . . . . .	63
4.2	Accuracy of various LMMs on <code>GlitchBench</code> . Numbers highlighted in ■ represent the average results of Q1 and Q2, which are the main results of the benchmark. Numbers related to Q3 serve as a visual perception test to measure the ability of models to report glitches in a relaxed manner. Numbers highlighted in ■ show the maximum agreement achievable with ground truth as perceived by Llama-2’s judgment (%). Numbers highlighted in ■ represent the results obtained from GPT-4V on glitch-free images. . . . .	67
4.3	Evaluating a subset of responses for comparing Llama-2 with human judgments: Llama-2 and humans exhibit moderate to substantial agreement on all models except for Fuyu. . . . .	68
4.4	Comparing accuracy of models on <code>GlitchBench</code> and other visual benchmarks — the bold numbers show the best model per benchmark (%)	74
5.1	Categories of questions in our dataset, along with a sample question for each category. . . . .	88

- 5.2 Performance (accuracy %) of Bunny on the evaluation set after fine-tuning on a mixture of data with various strategies. The *Weighted* strategy leads to better performance with smaller dataset sizes, but as size increases, all strategies perform similarly. We use a strategy similar to *Weighted* to train VIDEOGAMEBUNNY with 50K samples. . . . . 91
- 5.3 Average accuracy improvement (percentage point) for different sizes for each category . . . . . 93
- 5.4 Performance (accuracy %) of various models on the evaluation set. . . . . 93
  
- A.1 Breakdown of Performance for Different LMMs by Various Glitch Types (%) . . . . . 131
- B.1 Description of entries in the JSON structure . . . . . 170
- B.2 Comparison of VIDEOGAMEBUNNY and LLaVA-1.6-34b . . . . . 172

# List of Figures

2.1	Examples of visual and logical glitches in video games . . . . .	13
2.2	Overview of different vision models with different levels of text supervision. . . . .	15
2.3	The architecture of LLaVA [112], where a pre-trained image encoder extracts image features that are then projected into the latent space of a language model. This allows the language model to enable conversation based on the image. . . . .	16
3.1	🔗 Video identified by our approach with the bug query ‘ <i>A horse in the air</i> ’ for Red Dead Redemption 2. . . . .	20
3.2	Sample instances of game physics bugs. . . . .	22
3.3	(a) Overview of our gameplay video search approach. (b) Overview of our data collection process. . . . .	29
3.4	Relevant gameplay videos identified using our approach with bug queries in a game from the Assassin’s Creed franchise. Our method can effectively retrieve highly specific bugs using textual descriptions, without depending on specialized vocabulary or technical terms. . . . .	46
4.1	The image depicts a screenshot in which it rains inside a room. While the rain should be what is wrong with the image, GPT-4V fails to reason correctly and instead focuses on the color of Batman’s costume. Note that the ground truth is never presented as part of the prompt in our study. . . . .	55
4.2	Sample images from the <code>GlitchBench</code> showing glitches in various games with distinct styles. Samples (a)–(e) are captured from online videos, while sample (f) is generated inside the Unity game engine. . . . .	57
4.3	To evaluate a model’s response, we ask a judge (the Llama-2-70b-Chat model) to compare it semantically with the ground truth. . . . .	65
4.4	The performance (accuracy %) of all tested models on different categories of images in <code>GlitchBench</code> . . . . .	69

4.5	One of the several cases in which GPT-4V fails to detect a problem with facial features. . . . .	70
4.6	The image shows a basketball player with an unnatural, impossible elbow pose. GPT-4V fails to focus on small details such as body configuration and is unable to report this issue. . . . .	72
5.1	VIDEOGAMEBUNNY is a model specifically fine-tuned on video game content, enabling it to understand game contexts and respond to related questions more accurately. . . . .	76
5.2	Architecture overview of VIDEOGAMEBUNNY. An image input and a textual instruction are fed into the language model to produce a response. The image is passed through a separate pre-trained vision encoder and a projection layer to align the embedding space between the two models. 🔥 and ❄️ icons show trainable and frozen layers respectively . . . . .	82
5.3	Our dataset includes sample video game images that showcase a wide range of characters, environments, mechanics, camera viewpoints, and artistic styles. These styles vary from western to contemporary and futuristic, and from realistic to fantasy settings. . . . .	83
5.4	Overview of the dataset generation process and models used to generate each type . . . . .	84
5.5	Sample information extracted for the image-to-JSON dataset by Gemini-1.5-Pro. Each sample contains detailed information ranging from minor details to high-level descriptions, such as: <b>1</b> player inventory, <b>2</b> <b>3</b> details about the environment, <b>4</b> non-player characters, <b>5</b> the screenshot’s watermark, and <b>6</b> lighting. . . . .	86
5.6	Relative accuracy improvement (percentage points) of Bunny fine-tuned on different subsets of each dataset. The image-to-JSON dataset shows a strong positive trend, while the short captions dataset degrades performance. The best performance achieved in the experiment is highlighted in bold. . . . .	90
5.7	VIDEOGAMEBUNNY correctly utilizes information on the HUD to answer the question, while the basemodel Bunny and the larger model LLaVA-1.6-34b fail to provide a correct answer. . . . .	94
5.8	VIDEOGAMEBUNNY provides a detailed description of video game content in the image, while other models show inaccuracies (highlighted in <i>red</i> ). Responses are truncated to save space. . . . .	95

A.1	Samples for which models failed to reason about the content of the image and instead responded with <i>nothing</i> and similar phrases. . . .	133
A.2	GPT-4V not only fails to recognize the absence of the facial skin, but it also hallucinates that the character’s head is replaced with that of a chimpanzee. . . . .	135
A.3	GPT-4V provides a detailed description of the image, yet it fails to notice that the head is missing. The judge’s response truncated to save space. . . . .	136
A.4	GPT-4V can detect some details from the image and the franchises of video games, yet it cannot recognize that the hands of the character shown in the image are unnaturally stretched to reach the box. . . . .	138
A.5	LLaVA-1.5 can describe the image content and details, such as the character’s clothing, but it fails to notice that the character’s leg is bent in an unnatural way. . . . .	139
A.6	Qwen-VL can describe the image correctly, but it fails to understand the character’s body configuration and the unnatural hand positions. . . . .	140
A.7	GPT-4V starts by describing the image as a video game screenshot and then begins to read the text on the image. Regarding the actual content of the image, it fails to understand the content and hallucinates about a person pushing a photocopier. The model and judge’s response are truncated to save space. . . . .	141
A.8	While GPT-4V detects that the image is a screenshot of a first-person video game with a person holding a weapon, it fails to detect the floating towel; instead, it hallucinates about a mirror. . . . .	142
A.9	Sample hallucination made by InstructBLIP-7B. The model mentions the presence of a horse in front of a wooden horse. . . . .	143
A.10	While LLaVA-1.5-7B provides some details about the image, such as a person lying on the ground, it hallucinates the presence of a computer mouse and a person holding a book. . . . .	144
A.11	The screenshot displays a car floating in the air, but Fuyu-8B provides incorrect details and generates numerous hallucinations. All details are incorrect. . . . .	145
A.12	The screenshot shows a scene from The Witcher 3 game in which a horse has an unusual posture. MiniGPT-v2 provides incorrect details and generates numerous hallucinations. Almost all details are incorrect. . . . .	146
A.13	Sample image in which SPHINX hallucinates about a dog. However, some details about the image, such as the presence of a car, are correct. . . . .	147

A.14 Llama-2 is confused and accepts the wrong response. The response generated by LLaVA-1.5-13B hallucinates a floating sword, which is incorrect. However, Llama-2 matches “desert-like environment” with the ground truth: “A pixelated character is floating in the air.”. The model and judge’s response are truncated to save space. . . . .	148
A.15 MiniGPT-v2 produces a partial description of the image with some hallucination, leading Llama-2 to accept the wrong answer as a correct match with the ground truth. . . . .	149
A.16 While the InstructBLIP’s response contains some key elements such as crocodiles and wooden planks, it does not mention the clipping issue. InstructBLIP also hallucinates about crocodiles swimming in the water. However, Llama-2, acting as the judge, aligns with the theme of the ground truth in the InstructBLIP’s response and accepts the answer. . . . .	150
A.17 OtterHD thinks that the presence of soldiers and guns in the bedroom is an unusual aspect of the image. However, it fails to notice the clipping issue and mistakes it for a soldier kneeling on the ground. Llama-2, as the judge, acknowledges that the generated text and the ground truth have different wording, yet it completely fails to evaluate the meaning of the response and incorrectly accepts it. . . . .	151
A.18 Despite the fact that SPHINX’s response does not mention any problem with the water, Llama-2 matches SPHINX’s response with the ground truth and accepts it. . . . .	152
A.19 GPT-4V fails to detect a clipping glitch in which two cars of the same model and visually identical are placed on top of each other on a dirt road. The model and judge’s response are truncated to save space. . . . .	153
A.20 The screenshot shows a glitch in which a character is standing in a doorway frame while the door is closed, resulting in clipping with the door. GPT-4V fails to notice that the door is closed. The judge’s response is truncated to save space. . . . .	154
A.21 LLaVA-1.5 detects some individual objects from the image, for example, the car and its driver, rocks and bushes, but it does not understand the clipping issue happening between the car and rocks (or stone barrier according to the ground truth). . . . .	155
A.22 MiniGPT-v2 provides a detailed description of the image but fails to notice that two people are placed in a strange way and are clipping into each other on the bed. . . . .	156

A.23	The screenshot shows a person’s body and a coffee cup intersecting and clipping with the person’s hand. InstructBLIP mistakes the person for a mannequin and also fails to notice the coffee cup entirely . . . . .	157
A.24	Qwen-VL provides an inaccurate description of the image, including multiple hallucinations about a person standing on the rooftop. Ultimately, Qwen-VL fails to notice the helicopter clipping into the wall.	158
A.25	The screenshot shows two ambulance cars and a police car colliding and intersecting with each other. Two ambulance cars are moving in the same direction, following the street traffic, but the police car is rotated 90 degrees, as if it was blocking the road. However, both ambulances are colliding with the police car. GPT-4V fails to detect these clipping issues. The judge’s response is truncated to save space. . . . .	159
A.26	The screenshot displays an unusual setting where a person stands in a furnace with glowing fire. GPT-4V fails to correctly detect and understand the scene, mistaking the furnace for a thermal monitor. Responses are truncated to save space. . . . .	160
A.27	The screenshot show two people are driving an invisible car due to a rendering glitch. GPT-4V fails to notice the strange character postures, which suggest that they are participating in an animation where they are riding in a car. The judge’s response truncated to save space. . .	161
A.28	The screenshot shows a person dressed in a sniper suit floating in the air near the center of the image (above the crosshair). Detecting the floating person can be challenging for some users due to the pattern used in the sniper suit, the background palm tree, and the overall color of the environment. . . . .	162
A.29	In this image, the cat on the left side of the image is slightly above the ground and floating in mid-air. Due to the lighting conditions and distance of the cat from the camera, detecting the glitch is hard. . . .	163
A.30	In this image, there is a character smoking a cigarette on the right side, but due to a rendering glitch, the character is not rendered at all; only the cigarette is visible. Detecting the absence of the character can be challenging for some users. . . . .	163
A.31	The image shows a significant rendering glitch in which vertices and triangles of the object are completely corrupted. Describing what is wrong with this image can be challenging for some users as they do not know specific terminology. . . . .	164

A.32	The image shows a blood and gore setting in a zombie-related game, with zombie intestines all over the place. Some users fail to notice that the hands are reloading a gun, but the magazine is being put into the wrong part of the gun, resulting in a clipping glitch. . . . .	164
A.33	The image depicts a rendering glitch known as “Hall of Mirrors” or “ghosting”, which results in a trail of previously rendered frames appearing instead of a missing mesh or texture. While detecting that there are some issues with the image is easy for most users, using the correct terms can be challenging. . . . .	165
A.34	The roof has a low-resolution texture. . . . .	166
A.35	The ladder has a placeholder texture. . . . .	166
A.36	Part of the roof has a placeholder texture. . . . .	166
A.37	The carriage has a distorted mesh. . . . .	166
A.38	Part of the house structure has a placeholder texture. . . . .	167
A.39	The canopy structure has a low-resolution texture. . . . .	167
A.40	The barrel’s mesh is stretched and distorted. . . . .	167
A.41	The boat has a low-resolution texture. . . . .	167
B.1	Long caption generation with GPT-4V . . . . .	168
B.2	image-to-JSON data generation . . . . .	169
B.3	LLama-3-based data generation . . . . .	169
B.4	GPT-4o-based data generation . . . . .	170
B.5	Image distribution across games, with a median of 205 unique images per game. . . . .	170
B.6	Breakdown of improvement (percentage points) compared to the base-model Bunny, after fine-tuning it on different mixture strategies by different question categories. As we increase the size of the dataset, all categories show improvement, with <i>Anomalies and Glitches</i> showing the greatest gains. . . . .	171
B.7	VIDEOGAMEBUNNY provides a detailed description of the image . .	173
B.8	VIDEOGAMEBUNNY provides a detailed description of the image . .	174
B.9	VIDEOGAMEBUNNY provides a detailed description of the image with minor inaccuracies. Details that are incorrect are highlighted and emphasized in <i>red</i> . . . . .	175
B.10	VIDEOGAMEBUNNY hallucinates extra characters and a Batman utility belt in the image, which are not actually present. Details that are incorrect are highlighted and emphasized in <i>red</i> . . . . .	176

B.11 A sample showing that VIDEOGAMEBUNNY struggles to convert the image to text. . . . .	177
B.12 A sample showing that VIDEOGAMEBUNNY fails to provide a correct answer. While the word <i>RELOAD</i> is written in the center of the image, the query asks, <i>What is written on the left?</i> which should be answered with <i>nothing</i> . . . . .	178

# Chapter 1

## Introduction

The video game industry, a dominant force in global entertainment, is set for unprecedented growth. Projections indicate a potential market valuation of \$321 billion by 2026 [44, 158], highlighting its rapid expansion and innovative trajectory. This remarkable growth is driven by emerging technologies, evolving gaming experiences, and an expanding global player base.

The industry's future success promises far-reaching economic impacts, from job creation to technological advancements benefiting various sectors. However, developing immersive video games remains a complex endeavor, blending technical proficiency with creative artistry. This complexity often results in demanding schedules and potential developer burnout, necessitating more efficient development strategies.

Artificial Intelligence (AI) has demonstrated immense potential across sectors such as healthcare [124], agriculture [51], and finance [128], driving efficiency and growth. The video game industry is experiencing a similar revolution. AI advancements, particularly in natural language processing, are enhancing Non-Playable Characters (NPCs), making them more realistic and conversational [135]. Moreover, AI is accelerating game development, with tools like NVIDIA's *Neuralangelo* [100] simplifying the creation of high-fidelity 3D models from real object images. However, the potential of AI in Quality Assurance (QA), a crucial component of game development, remains largely unexplored.

This thesis focuses on leveraging the potential of AI, specifically *foundation models*, to enhance the QA processes in gaming companies.

The statement of this thesis is that AI models can assist in detecting 50% of common video game glitches and have the potential to enhance quality assurance processes, alleviate developer burden, and ultimately improve game quality.

## 1.1 Foundation models – A new paradigm

Foundation models [21] represent a significant paradigm shift in AI. These large, general-purpose machine learning models, trained on vast and diverse datasets, serve as the basis for numerous downstream applications. Unlike traditional domain-specific AI systems, foundation models demonstrate remarkable versatility across a wide range of tasks with minimal modifications. The power of these models lies in their ability to perform multiple tasks by conditioning on a small amount of task-specific data at inference time, a capability enabled by their extensive pre-training [152, 24, 39]. This approach has garnered significant attention and investment from major technology corporations, including Microsoft, NVIDIA, Meta, and Google. Prominent examples of foundation models include:

- The GPT-4 series [137, 138, 136] from OpenAI is capable of generating human-like textual conversations.
- The Stable Diffusion series [160, 42] from Stability transforms textual inputs into realistic imagery.
- The Luma Dream Machine [118] is capable of creating realistic videos from text and image inputs.

Foundation models offer several distinctive advantages compared to transitional AI models:

- **Flexibility:** These models can be adapted to solve diverse problems, eliminating the need for separate models for individual tasks [21].
- **Generalizability:** By leveraging large datasets and model sizes, foundation models can capture complex patterns and effectively apply their knowledge to new domains [24].
- **Broad Capabilities:** Their versatility extends across various fields, including natural language processing, computer vision, and robotics [156].
- **Potential for Cost-Effectiveness:** While initial development costs can be high, the ability to apply these models to multiple tasks can potentially lead to long-term cost savings. Additionally, emerging technologies like LoRA (Low-Rank Adaptation) [68] are making it more feasible for smaller teams to utilize and fine-tune these models without enormous upfront costs.

These properties of foundation models raise an intriguing question: How well do these models perform in reasoning about video game content, and what are their potentials of being used in industrial settings to enhance existing pipelines?

## 1.2 Foundation models for video game quality assurance

The potential of foundation models to reason about diverse content and their applicability in various industrial settings is a topic of growing interest [21]. One particularly promising area for the application of these models is the video game industry. The versatility and advanced capabilities of foundation models make them well-suited to address various challenges in game development, from content creation to testing and quality assurance. In the following section, we will explore how these models are being leveraged specifically in the context of video game quality assurance, demonstrating their potential to revolutionize traditional development processes.

The potential of foundation models, exemplified by GPT-4 [137], is particularly evident in the creation of compelling in-game interactions. By facilitating dynamic and more lifelike dialogues for NPCs, these technologies contribute to a more authentic and engaging gaming experience [140]. However, the impact of foundation models extends beyond player immersion, transforming various aspects of the game development process. These AI-driven technologies expedite asset creation [149, 67, 220, 30, 149, 67], and also automate tasks such as code generation [98, 196]. This broad spectrum of applications underscores the potential of foundation models to enhance game development significantly.

In the game development lifecycle, QA presents a unique opportunity for the application of foundation models. This critical aspect of game development traditionally relies heavily on human testers, resulting in significant resource demands due to its labor-intensive nature. Human testers are typically required to repeatedly play the same game scenes to ensure both playability and consistently high quality throughout the development process.

While the use of game-playing bots, facilitated by recent advancements in reinforcement learning [77, 171], is an established practice in game development, they are not universally applicable to all game-testing processes. Effective game testing necessitates human-like responses to events, visuals, and interactions within a game environment. For instance, a typical game bot might overlook an unexpected event, such as a *flying horse* in a game scene, due to its lack of common sense reasoning capabilities.

The limitations of both human testers and traditional game-playing bots in QA processes highlight the need for more advanced, versatile solutions. Foundation models, with their broad capabilities and human-like reasoning, have the potential to bridge this gap. These models could potentially combine the efficiency of automated testing with the nuanced understanding of human testers, enabling more comprehensive and insightful QA processes. This potential application of foundation models in

game QA presents an exciting area for further research and development.

## 1.3 Thesis objectives

The central objectives of this thesis are as follows:

1. To explore and advance the application of foundation models and computer vision techniques in video game quality assurance, particularly in the classification and retrieval of existing glitches.
2. To investigate and improve the capabilities of large multimodal models in comprehending and interpreting video game content, with a focus on enhancing visual quality assurance in game development.

To achieve our objectives, we conducted three studies. The first study primarily addresses the first objective by exploring novel approaches to glitch classification. This study focuses on scenarios where the types of glitches being looked for are predefined. Studies 2 and 3 focus on the second objective, evaluating and improving the performance of large multimodal models in understanding video game content in a more open-ended manner. These studies investigate the models' ability to identify and describe potential glitches without being limited to a predefined set of glitch categories.

### **Research Study 1: Searching bug instances in gameplay video repositories (Chapter 3)**

**Motivation:** Video game glitches present a unique challenge in game QA due to their diverse and unpredictable nature. Traditional image classifiers, in particular, have serious limitations when dealing with such diverse scenarios and are hard to maintain. Recognizing this limitation, we explore the potential of leveraging image-text pre-trained models, specifically CLIP, for video game glitch retrieval tasks based

on given textual queries without relying on any specific training dataset. By exploring the capability of these models to match textual descriptions of glitches with visual input, we develop a more flexible and adaptable system for retrieving a wide range of video game anomalies from a repository of gameplay videos.

**Findings:** Image-text models like CLIP demonstrate remarkable effectiveness in identifying and matching video game glitches based on textual descriptions, as revealed by our two-phase study. We began with an analysis of eight diverse commercial video games, followed by an in-depth investigation of a single title. Our comprehensive study showcased the models’ ability to retrieve relevant videos from large repositories using only textual input, highlighting a significant advancement in video game content analysis in a zero-shot manner, meaning the model has never been trained for new glitch types. Our findings show that image-text models can successfully generalize to video game content, paving the way for novel approaches in various video game-related tasks and analyses.

## **Research Study 2: GlitchBench: Can large multimodal models detect video game glitches? (Chapter 4)**

**Motivation:** The rapidly evolving landscape of large multimodal models, exemplified by GPT-4V, presents exciting opportunities for creating visual assistance across diverse domains, from everyday applications to specialized fields such as healthcare. Recognizing this potential, our study seeks to explore the capabilities of these models in the context of video game quality assurance, specifically focusing on glitch detection in screenshot images. To assess the efficacy of both proprietary and open-source models in describing, detecting, and comprehending visual glitches, we introduce **GlitchBench**, a new benchmarking dataset. Through extensive evaluation of 11 diverse models on **GlitchBench**, we uncover their strengths, limitations, and overall potential in automating and enhancing the critical process of visual quality control

in game development.

**Findings:** Our study reveals that GPT-4V demonstrates superior performance in single-image glitch detection, achieving an accuracy of 64.9%. This result not only sets a new benchmark but also indicates significant potential for advancement, with a 35% headroom for improvement still possible for future models. Notably, open-source models lag considerably behind, with the best performer, LLaVA [111], reaching only 30.5% accuracy. This highlights a substantial performance gap between proprietary and open-source models.

Comparing open-source models with GPT-4V across various benchmarks and `GlitchBench` reveals differences in generalization and practical application. While LLaVA achieves high accuracy on many benchmarks, it still performs extremely poorly compared to GPT-4V on `GlitchBench`. This suggests a need for comprehensive benchmarks that assess a model’s capability across diverse tasks and its ability to generalize, rather than focusing on narrow domains that may not translate to broader applications.

### **Research Study 3: VIDEOGAMEBUNNY: Towards vision assistants for video games (Chapter 5)**

**Motivation:** Study 3 reveals a significant gap between open-source and closed-source models in understanding and comprehending video game content. Motivated by this, we investigate how fine-tuning a foundation model with game-specific data can improve its understanding of game content. First, we introduce a dataset containing 389,565 image-instruction pairs, which include image captions, question-answer pairs, and a JSON representation of 16 elements of 136,974 images. Then we explore fine-tuning an open-source model on this data using various mixture strategies to improve its performance on game understanding tasks.

**Findings:** In this study, we introduce a model called `VIDEOGAMEBUNNY`, which outperforms larger state-of-the-art open-source models in video game understanding tasks. While developing this model, we observe and identify several key findings:

We find that starting with an open-source model and fine-tuning it on a specially crafted dataset of video game images can improve performance over the baseline. Our exploration of different training datasets suggests that a structured format like image-to-JSON is the most effective in improving model performance. Conversely, short captions are not effective and may even degrade the model’s performance.

Our results indicate that the effectiveness of data type mixture strategies depends on the dataset size. With smaller dataset sizes, a weighted strategy that favors higher-quality datasets is more effective than other strategies. However, as the dataset size increases, we observed that all strategies perform similarly, with no statistically significant differences between various data mixtures.

Our model `VIDEOGAMEBUNNY`, trained on 50K high-quality video game-specific instructions, shows an improvement of **+11.8** over the baseline model and also outperforms LLaVA-1.6-34b, which has  $4.2\times$  more parameters, on a video game understanding benchmark.

## 1.4 Thesis outline

The remainder of this thesis is organized as follows:

- Chapter 3 presents an investigation of image-text models, particularly CLIP, in detecting and analyzing video game glitches. This chapter explores the potential of these models to identify a wide range of visual glitches in video games without relying on extensive, predefined training datasets.
- Chapter 4 presents our research on the efficacy of large multimodal models in detecting video game glitches. This chapter introduces `GlitchBench`, a new

benchmarking dataset, and provides a comprehensive evaluation of both proprietary and open-source models in the context of video game quality assurance.

- Chapter 5 presents a study where we develop `VIDEOGAMEBUNNY`, a video game-specific large multimodal model. This chapter explores different dataset types and mixture strategies to improve model performance in understanding and interpreting video game content.
- Chapter 6 concludes the thesis by summarizing our key findings, discussing their limitations and implications for the field of AI in video game analysis, and suggesting directions for future research.

# Chapter 2

## Background

### 2.1 Video game glitches

A video game glitch can be defined as an unintended anomaly in a game’s behavior or presentation, resulting from errors in the game’s code, design, or implementation. Glitches manifest as unexpected or erroneous events that deviate from the game’s intended functionality or aesthetics. These anomalies can range from minor visual quirks to significant disruptions in gameplay mechanics. It is worth noting that in common usage, the terms “glitch” and “bug” are often used interchangeably in the context of video games.

#### 2.1.1 Notable examples of glitches

The history of video games is filled with memorable glitches that have become famous within the gaming culture. These examples illustrate how glitches can range from amusing quirks to game-changing phenomena:

1. **The Skyrim bee bug** [183]: In the opening sequence of the Elder Scrolls V: Skyrim game, where the player rides in a cart with other prisoners, some players encountered a strange issue. The cart would sometimes unexpectedly fly off the path and launch into the sky. After extensive investigation by the game development team, it was discovered that this was caused by a collision issue with an in-game bee. A bug fix intended to make bees collectable inadvertently

caused this issue. If a bee happened to cross paths with the cart during the intro sequence, it would cause the cart to be flung into space, taking everything inside with it.

2. **Cyberpunk 2077's launch issues** [35]: When released in 2020, Cyberpunk 2077 was plagued with numerous bugs and performance issues, particularly on last-gen consoles. These ranged from visual glitches and AI malfunctions to game-breaking bugs that prevented quest completion. This case highlighted the challenges of developing complex open-world games and the importance of thorough testing across different platforms.
3. **Assassin's Creed Unity's disappearing face glitch** [8]: In this action-adventure game, a glitch caused characters' faces to disappear, leaving only floating eyes and teeth. This visual glitch became widely shared online, overshadowing the game's release and highlighting the potential for a glitch to significantly impact a game's reception and public image.

### 2.1.2 The challenge of glitch detection in game development

In the game development lifecycle, quality assurance (QA) plays a crucial role in identifying and reporting bugs and glitches. Game QA is a repeatable process that involves systematic testing of the game at various stages of development to ensure it meets quality standards and functions as intended. QA testers use a variety of techniques, including playtesting, stress testing, and automated testing tools, to uncover issues that may have been overlooked during development.

While game testing encompasses a wide range of potential issues, including performance problems, AI behavior verification, and various types of bugs, this thesis focuses specifically on two categories of glitches that present unique challenges in detection and resolution: visual glitches and common-sense glitches. These types of glitches stand out due to their unique nature and the way they are perceived and

identified. They are distinct from other software-related bugs in that they often require direct observation of the game’s output and, in some cases, reasoning about the game’s context to identify them as anomalies.

**Visual glitches** are anomalies in the game’s graphical presentation. These can include rendering issues, texture problems, or unexpected visual artifacts. For example, a visual glitch might cause the external walls of a building to render incorrectly or not at all, allowing the player to see inside structures that should be opaque. Such glitches are immediately apparent to the player, as they create a clear discrepancy between what is seen and what is expected in the game world. Figure 2.1a illustrates this type of visual glitch, where building textures are not rendered correctly, resulting in missing or incomplete surfaces.

**Common sense glitches**, on the other hand, involve situations or behaviors within the game that, while perhaps technically functioning as programmed, violate the logical expectations set by the game’s context or real-world analogues. A prime example of this would be a shark swimming through the air as if it were in water. While the shark’s movement might be executing correctly according to the game’s code, it fundamentally contradicts the player’s expectations and the implied rules of the game world. This exact scenario is depicted in Figure 2.1b, which shows a shark floating in mid-air, contradicting both the physics of the real world and the game’s rules.

These two categories of glitches present unique challenges in detection, often requiring a combination of visual understanding and contextual reasoning that goes beyond traditional software testing methodologies. This thesis focuses on utilizing recent advances in vision-language models to identify and address these specific types of glitches, with the aim of improving the overall quality and immersion of video game experiences.



(a) A visual glitch where building textures are not rendered correctly, resulting in missing or incomplete surfaces.



(b) A common sense glitch showing a shark floating in mid-air, defying the expected underwater or on-land environment.

Figure 2.1: Examples of visual and logical glitches in video games

## 2.2 Vision-language models

Vision Language Models (VLMs) represent a significant advancement in artificial intelligence, aiming to bridge the gap between computer vision and natural language

processing. Early approaches to combining vision and language often treated these domains separately, with dedicated models for each modality. However, with the introduction of the transformer [191] architecture, which brings a high degree of uniformity in model design across various domains and tasks, coupled with the increasing availability of large-scale multimodal datasets, we are now witnessing the rise of multimodal models.

A pivotal moment in this evolution came with the introduction of CLIP [151] (Contrastive Language-Image Pre-training) by OpenAI. CLIP represented a departure from previous approaches by using a dual-encoder architecture: An image encoder (based on a vision transformer [40] or ResNet [60]), and a text encoder (based on a transformer architecture [151]).

CLIP’s key innovation lay in its training approach, using contrastive learning on a massive dataset of 400 million image-text pairs. This allowed CLIP to learn a joint embedding space for images and text, enabling zero-shot transfer to various vision tasks. However, it is important to note that CLIP is an encoder-only model, focusing on representation learning rather than generation.

Building on CLIP’s success, Google introduced LiT [218] (Locked-image Text Tuning). LiT maintained the encoder-only architecture but introduced a novel training strategy. It “locked” the weights of a pre-trained image encoder while fine-tuning the text encoder on image-text datasets. This approach allowed LiT to leverage high-quality image representations from models pre-trained on large vision datasets while adapting to potentially noisy image-text data.

While CLIP and LiT made significant strides in vision-language representation learning, they were limited in their ability to generate text or perform more complex language understanding tasks. This limitation led to the development of more sophisticated architectures that incorporated full-fledged language models.

BLIP (Bootstrapping Language-Image Pre-training) marked a transition towards more comprehensive vision-language architectures. BLIP introduced a multi-modal

mixture of encoder-decoder (MED) design, which included: an image encoder, a text encoder and a text decoder. This architecture allowed BLIP to not only perform image-text matching but also generate captions and engage in more complex language tasks. However, the text decoder in BLIP was not a full pre-trained language model, but rather a more lightweight decoder trained specifically for the task. Figure 2.2 provides an overview of the different models and their architectures.

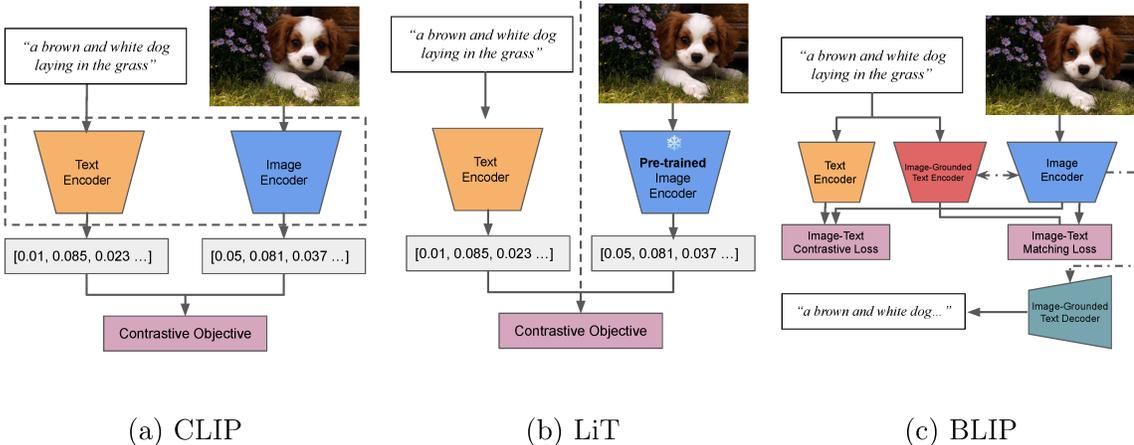


Figure 2.2: Overview of different vision models with different levels of text supervision.

The next major advancement came with models like LLaVA [112] (Large Language and Vision Assistant), which represent one of the most popular architectures in VLMs. These models take a significant step forward by incorporating fully pre-trained language models as decoders. LLaVA, for instance, fine-tunes a large language model (e.g., LLaMA [186]) with visual features extracted from a pre-trained vision encoder. Figure 2.3 shows the architecture of LLaVA. This architectural choice brings several advantages:

1. The use of a full pre-trained language model as a decoder allows these VLMs to leverage the broad language understanding and generation capabilities developed through large-scale language model pre-training.
2. This architecture enables more sophisticated multi-modal reasoning, as the language model can process both the visual information (as encoded by the visual

encoder) and textual inputs.

3. LLaVA can perform a wide range of vision-language tasks, from image captioning to visual question answering, while maintaining the powerful language understanding and generation capabilities of large language models.

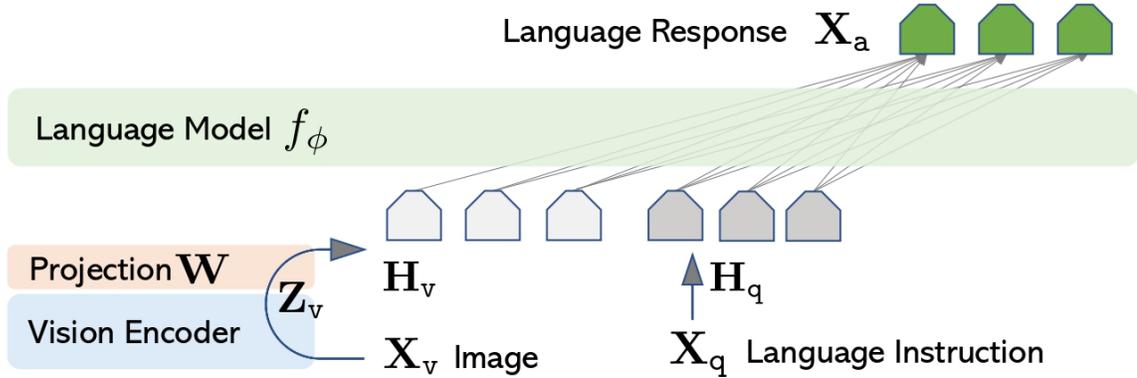


Figure 2.3: The architecture of LLaVA [112], where a pre-trained image encoder extracts image features that are then projected into the latent space of a language model. This allows the language model to enable conversation based on the image.

The transition from encoder-only models like CLIP and LiT to the more recent architectures exemplified by LLaVA represents a significant evolution in the field of vision-language modeling. These newer models offer several advantages:

1. **Enhanced generation capabilities:** The incorporation of full language models allows for more natural and diverse text generation in response to visual inputs.
2. **Improved multi-modal reasoning:** These models can perform more complex reasoning tasks that require integrating information from both visual and textual modalities.
3. **Greater flexibility:** The use of pre-trained language models allows these VLMs to adapt more easily to a wide range of downstream tasks with minimal fine-tuning.

4. **Zero-shot and few-shot learning:** Leveraging the capabilities of large language models, these VLMs can often perform new tasks with little or no task-specific training data.

In conclusion, the evolution of VLMs, from encoder-only architectures to those incorporating fully pre-trained language models, represents a significant step towards more capable and flexible AI systems that can seamlessly integrate visual and linguistic information. These advancements open up new possibilities for applications in areas such as visual question answering, image-grounded dialogue systems, and multi-modal content generation. As the field continues to evolve, we can expect to see further refinements in how visual and linguistic information is integrated and processed.

## Chapter 3

# Searching bug instances in gameplay video repositories

**Bibliographic information:** This chapter was published as: Taesiri, Mohammad Reza, , Finlay Macklon, Sarra Habchi, and Cor-Paul Bezemer.. “Searching Bug Instances in Gameplay Video Repositories” in *IEEE Transactions on Games* (2024) [177]. This is a follow-up work, in collaboration with our industry partner, which was originally published as: Taesiri, Mohammad Reza, Finlay Macklon, and Cor-Paul Bezemer. “CLIP Meets GamePhysics: Towards Bug Identification in Gameplay Videos Using Zero-Shot Transfer Learning” [176] in the *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)* (2022).

Gameplay videos offer valuable insights into player interactions and game responses, particularly data about game bugs. Despite the abundance of gameplay videos online, extracting useful information remains a challenge. This chapter introduces a method for searching and extracting relevant videos from extensive video repositories using English text queries. Our approach requires no external information, like video metadata; it solely depends on video content. Leveraging the zero-shot transfer capabilities of the Contrastive Language-Image Pre-Training (CLIP) model, our approach does not require any data labeling or training. To evaluate our approach, we present the `GamePhysics` dataset, comprising 26,954 videos from 1,873 games that were collected from the GamePhysics section on the Reddit website. Our approach shows promising results in our extensive analysis of simple and compound queries,

indicating that our method is useful for detecting objects and events in gameplay videos. Moreover, we assess the effectiveness of our method by analyzing a carefully annotated dataset of 220 gameplay videos. The results of our study demonstrate the potential of our approach for applications such as the creation of a video search tool tailored to identifying video game bugs, which could greatly benefit Quality Assurance (QA) teams in finding and reproducing bugs. The code and data used in this chapter can be found at <https://zenodo.org/records/10211390>

### 3.1 Introduction

Video game development is a highly complex process. There are many unique challenges when applying general software engineering practices to video game development [147, 132, 172, 143, 90], including challenges in game testing. Manual testing is a widely accepted approach to game testing [146, 190, 141], however, this manual process is slow and error-prone, and most importantly, expensive. On the other hand, it is challenging to automate game testing [89, 141, 163] due to the unpredictable outputs of video games. Despite progress towards automated game testing methods [38, 105, 190, 189] and tools [76, 229, 17, 144], new approaches to game testing must be researched.

The difficulty of game testing due to the unique nature of games calls for unique testing methodologies as well. For example, we could leverage the visual aspect of games in the testing process. Having a gameplay video is very helpful when trying to reproduce a bug in the development environment for further analysis, as bug reports often contain incomplete information [19]. The ability to search a large repository of gameplay videos with a natural language query would be useful to help reproduce such bug reports. For example, in the game development domain, a bug report might state “a horse is flying in the air” (Figure 3.1) without a screenshot or video to show what is actually happening. A gameplay video search would allow game developers to find example instances of a specific bug in the pile of gameplay videos from their



Figure 3.1: [🔗](#) Video identified by our approach with the bug query ‘*A horse in the air*’ for Red Dead Redemption 2.

playtesting sessions or the internet (e.g., YouTube, Twitch).

Despite containing rich information, the challenges related to video parsing and understanding mean that gameplay videos are difficult to utilize. Manually identifying bug instances is time-consuming, and there is limited prior research on automatic methods for mining large repositories of gameplay videos [102, 120].

In this paper, we address the challenges of extracting useful information from large repositories of gameplay videos. We propose an approach for mining gameplay videos using natural language queries by leveraging the Contrastive Language-Image Pre-Training (CLIP) model [151] to identify similar text-image pairs without any additional training (i.e., zero-shot prediction). We leverage CLIP for videos by pre-processing the video frame embeddings and use Faiss [75] to perform a fast similarity search for the pairs of text queries and video frames. In our approach, we present two methods to aggregate the similarity scores of each text-image pair to identify relevant videos. Figure 3.1 shows an example of a video that was identified by our approach when searching videos from the Red Dead Redemption 2 game using the bug query

‘*A horse in the air*’. The primary application of our approach is as a gameplay video search engine to aid in reproducing game bugs.

To evaluate our approach, we collected and prepared the `GamePhysics` dataset, consisting of 26,954 gameplay videos that contain outstanding game physics examples (including many bugs). We first did a preliminary evaluation of our approach with sets of simple queries, and compound queries, to show that our approach can identify objects and (bug-related) events in large repositories of gameplay videos. Our approach and the preliminary evaluation were published in our original study at the Mining Software Repositories conference [176]. In this chapter, we extend our original work with an in-depth evaluation of our approach on a commercial game from our industry partner.

The main contributions of our chapter are as follows:

- We propose an approach to search for objects and events in gameplay videos using a natural language text query. (Sec. 3.4)
- We present the `GamePhysics` dataset <sup>1</sup>, consisting of 26,954 gameplay videos from 1,873 games. (Sec. 3.5)
- We demonstrate the promising performance of our approach in identifying game physics bugs through 2 experiments with simple and compound queries in Sec. 3.6 and present the results in Sec. 3.6.8.
- We extend our original study [176] by conducting an extensive evaluation of our method on a game from our industry partner in which we compare several variations of CLIP. (Sec. 3.7)

The remainder of our chapter is structured as follows. Section 3.2 motivates our study by providing relevant background information. Section 3.3 discusses related work. Section 3.4 presents our approach to mining large repositories of gameplay

---

<sup>1</sup><https://huggingface.co/datasets/taesiri/GamePhysics>



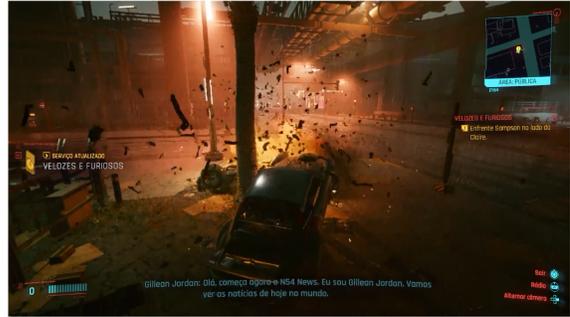
(a) [🔗](#) Bug in Grand Theft Auto V. Car stuck in a tree after colliding.



(b) [🔗](#) Bug in The Elder Scrolls V: Skyrim. Dragon stuck in the ground.



(c) [🔗](#) Bug in Red Dead Redemption 2. Incorrect sitting animation.



(d) [🔗](#) Bug in Cyberpunk 2077. Cars stuck together after colliding.

Figure 3.2: Sample instances of game physics bugs.

videos. Section 3.5 discusses collecting and pre-processing the `GamePhysics` dataset. Section 3.6 details the experimental setup of our preliminary evaluation, and Section 3.6.8 presents the results. Section 3.7 gives a detailed analysis of our proposed method in a controlled (industrial) environment and compares it with several alternative approaches. Section 3.8 provides discussion and insights on limitations and the performance of our approach. Section 3.9 addresses threats to validity, and Section 3.10 concludes the chapter.

## 3.2 Motivation and background

### 3.2.1 Video game (physics) bugs

In this chapter, we are interested in a specific category of bugs in video games that we call ‘game physics’ bugs. Game physics bugs are not necessarily related to an inaccu-

rate physics simulation. Many of these bugs are related to the faulty representation of game objects due to an error in the internal state of that object. A few sample instances of game physics bugs can be seen in Figure 3.2. In Figure 3.2a, a bug from Grand Theft Auto V related to object collisions is shown. Figure 3.2b shows a bug from The Elder Scrolls V: Skyrim, related to object clipping. In Figure 3.2c, a bug from Red Dead Redemption 2 related to ragdoll poses can be seen. Figure 3.2d shows a bug from Cyberpunk 2077, related to object collisions. Identifying game physics bugs is challenging because we need to be able to extract specific, high-level (abstract) events from the gameplay videos, that are often similar to correct behavior.

### 3.2.2 Challenges in mining gameplay videos

Until now, it has been challenging to extract valuable information from large repositories of gameplay videos. Identifying bug instances by manually checking the contents of gameplay videos is time-consuming [102]. Therefore, automatic methods for mining gameplay videos are required. The only existing approach for automatically extracting events from gameplay videos requires manual data labeling (and the training of new models) [120], which itself is time-consuming. Therefore, an effective method for extracting valuable information from gameplay videos should be able to automatically analyze the video contents without requiring manual data labeling.

### 3.2.3 Contrastive learning and zero-shot transfer

While there are many approaches toward zero-shot learning, we are interested in assessing the zero-shot performance of pre-trained contrastive models. Contrastive learning is a machine learning technique in which the goal is to learn a representation of inputs such that similar items stay close to each other in the learned space, while the dissimilar items are far away [15, 23]. In recent years, contrastive learning has been one of the key drivers in the success of self-supervised learning methods and has been used for zero-shot transfer learning [28, 55, 82, 151]. Zero-shot learning tackles

a family of problems in machine learning by letting an algorithm solve a task without having a training set for that specific task [85, 86]. To illustrate this idea, suppose that a person has never seen a zebra before. If we give them a detailed description of a zebra (e.g., an animal similar to a horse, but with black-and-white stripes all over their bodies), that person can identify a zebra when they see one.

### 3.2.4 The Contrastive Language-Image Pre-Training (CLIP) model

One contrastive model that has proven zero-shot transfer capabilities is the Contrastive Language-Image Pre-Training (CLIP) model [151], which can leverage both text and image inputs together. We decided to use CLIP in our original study because of its multimodal capabilities and the size of its training dataset. CLIP consists of two parts: a text encoder, and an image encoder. These two parts work individually, and they can accept any English text and image as input. When an encoder of this model receives an input, it will transform it into an embedding vector. These embedding vectors are high-level features that are extracted by the network, representing the input. More specifically, these embedding vectors are how the neural network represents, distinguishes, and reasons about different inputs. Both encoders of this model will produce vectors of the same dimension for image and text inputs. Not only do these vectors have the same dimension, but they are also in the same high-dimensional feature space and are therefore compatible with each other. For example, the embedding vector of the text ‘an apple’ and the embedding vector of an image of an apple are very close to each other in this learned space. CLIP was pre-trained on over 400 million pairs of images and text descriptions that were scraped from the internet and has several different backbone architectures: RN50, RN101, RN50x16, ViT-B/32, ViT-B/16, etc. The models with ‘RN’ in their name are ResNet-based [60] models using traditional convolutional layers, while the ‘ViT’ models are based on vision transformers [40].

### 3.3 Related work

Event extraction from video content is of special importance for various data mining tasks [123, 148]. Only two prior studies have explicitly explored automatic approaches for mining gameplay videos, with varying success. Lin et al. showed that using metadata (such as keywords) to identify YouTube videos that contain video game bugs is feasible [102], but our approach looks at the video contents, which Lin et al. do not take into account. Our approach is more useful for game developers, as we can identify objects and (bug-related) events within gameplay videos and do not rely on metadata. Luo et al. proposed an approach for automatic event retrieval in e-sport gameplay videos that requires manual data labeling, a fixed set of classes (events), and the training of new models [120]. Our approach is more robust and easier to set up, as we can search gameplay videos with any English text query to identify specific objects and events without performing manual data labeling. Zhang et al. [224] investigated the retrieval of specific moments in narrative-driven games using natural language queries that semantically match both the auditory and visual content of scenes. In contrast, our approach focuses on video game bugs and does not rely on audio information. Furthermore, our approach works with a broader range of games.

Although there is limited prior work on mining large repositories of gameplay videos, there are several studies that propose approaches to automatically detect graphics defects in video games. One of the earliest approaches for automated detection of graphics defects was published in 2008, in which a semi-automated framework was proposed to detect shadow glitches in a video game using traditional computer vision techniques [133]. Recent studies have utilized convolutional neural networks in their approach to automatically detect a range of graphics defects [38, 105, 175, 27]. Instead of detecting graphics defects, our work is concerned with the automatic identification of game physics bugs in gameplay videos.

Tuovenen et al. leveraged the visual aspect of games through an image-matching approach to create a record-and-replay tool for mobile game testing [189]. Our approach leverages the visual aspect of games in a different way; instead of recording tests through gameplay, we automatically identify bugs in gameplay videos.

Some studies have proposed approaches for the automated detection of video game bugs through static or dynamic analysis of source code. Varvaressos et al. proposed an approach for runtime monitoring of video games, in which they instrument the source code of games to extract game events and detect injected bugs [190]. Borrelli et al. proposed an approach to detect several types of video game-specific bad smells, which they formalize into a tool for code linting [22]. Our approach differs as we do not require access to the source code of games; instead, we identify video game bugs based solely on the contents of gameplay videos.

In addition to related work on automatic bug detection for video games, there exists a wide range of work that leverages recent advancements in deep learning to provide new tools and techniques that address problems faced by game developers. Several studies have sought to make AI methods accessible in the video game development and testing cycle, either through the game’s internal state, raw pixels or through a high-level neural network-based representation [81, 187, 122]. Some studies have proposed approaches to accompany a game designer through the creation process of a game by providing suggestions and explanations to the designer [59, 58, 80]. Other studies have incorporated reinforcement learning and evolutionary methods to create AI agents that can automatically play games [77, 18, 194]. These AI agents can be further employed to perform automated game testing sessions [161, 53, 229, 48]. Our work is different from those listed above, as we focus on assisting game developers by providing an approach to efficiently search large repositories of gameplay videos to find bug instances.

## 3.4 Our approach

To assist with the detection and analysis of game bugs, we propose an approach that quickly and effectively searches a large repository of gameplay videos to find a specific object or event in a particular game. For creating such a powerful search system, one could utilize a traditional supervised classification technique. However, any supervised classification method needs a training dataset, a test dataset, and a fixed number of classes. Maintaining these datasets and labeling each sample is demanding and labor-intensive. Conversely, CLIP provides zero-shot transfer learning capabilities that allow us to develop an approach to automatically mine gameplay videos while avoiding the aforementioned issues. Figure 3.3a shows an overview of our approach.

### 3.4.1 Encoding video frames and the text query

Our approach accepts a set of videos and any English text query as inputs. We first extract all frames from each video and then use CLIP to transform our input text query and input video frames into the embedding vector representations described in Section 3.2.4. We selected CLIP because it is flexible enough to accept any arbitrary English text as a query and compare it with a video frame, without any additional training.

### 3.4.2 Calculating the similarity of embeddings

As well as avoiding manual data labeling, our approach avoids depending upon any extra information, such as metadata, to search gameplay videos. Instead, we are able to calculate similarity scores solely based on the contents of the video frames and the text query. The similarity score in our problem is the distance between an embedding vector representing a text query and another embedding vector representing a video frame. To calculate similarity scores for the pairs of embedding vectors, we opted for cosine similarity, a widely-used similarity metric [192, 215, 43, 193]. We require

an exhaustive search to calculate the similarity score of the text query with each individual frame in each input video. The performance of an exhaustive search will decrease inversely with an increasing number of videos in a repository. To combat this, we use Faiss [75] to conduct an efficient similarity search.

### 3.4.3 Aggregating frame scores per video

Although CLIP is designed to accept text and images as inputs, we can leverage CLIP for videos by treating each video as a collection of video frames (i.e. a collection of images). To identify specific events that could occur at any moment in a gameplay video, we cannot subsample the video frames as suggested in the original CLIP, because due to the richness of events in a single gameplay video, skipping any part of the video may lead to information loss and inaccurate results. Therefore, we perform a similarity search on all frames of all videos by comparing each individual video frame with the target query text, and we subsequently aggregate the similarity scores across each video. Below we detail the design of two different methods for aggregating the video frame similarity scores for each gameplay video. Our approach supports the two aggregation methods without the need to re-calculate the similarity scores.

#### **Aggregating frame scores using the maximum score**

Our first score aggregation method ranks videos based on the maximum similarity score across all frames belonging to each video. This method is highly sensitive, as a single frame with high similarity can lead to an entire video being identified as relevant to the query.

#### **Aggregating frame scores using the similar frame count**

In the second score aggregation method, we begin by ranking all frames of the input videos based on their similarity scores with the text query. Then, we select a predefined number (the *pool size* hyperparameter) of the highest-ranked frames across all

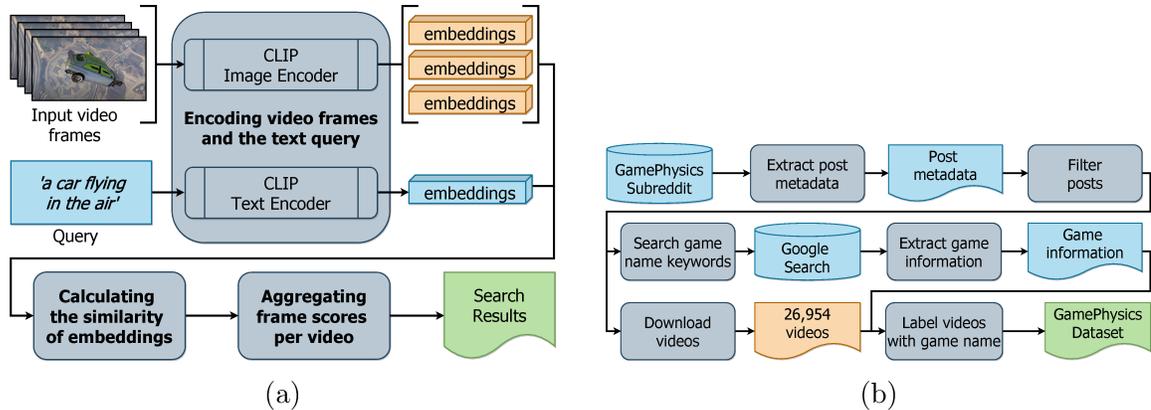


Figure 3.3: (a) Overview of our gameplay video search approach. (b) Overview of our data collection process.

videos. Finally, we count the number of frames per video within this pool of highest-ranked frames. This method is less sensitive than our first aggregation method, as identified videos must have multiple frames that are among the most similar to the input text query. We selected 1,000 as the default pool size value in our study.

## 3.5 Preparing the GamePhysics dataset

### 3.5.1 Collecting the GamePhysics dataset

Developing and testing a new machine learning system requires a dataset. Unfortunately, there is no such dataset for gameplay bugs. To this end, we present the **GamePhysics** dataset, which consists of **26,954** gameplay videos collected from the GamePhysics subreddit. An overview of our data collection process can be seen in Figure 3.3b.

#### Extracting post metadata and downloading videos

To collect the data, we created a custom crawler that uses both the official Reddit and the `PushShift.io` [13] APIs. We use the `PushShift.io` API to get high-level information about each submission in the GamePhysics subreddit. After obtaining high-level data, we use Reddit’s official API to update the scores and other metadata of each submission. For downloading video files, we combine `youtube-dl` and `aria2c`

to extract links and download them.

### **Filtering posts**

We applied several filters to our dataset during the data-collecting process to remove spam posts, low-quality content, and outliers. There are several spam posts in the GamePhysics subreddit, and these posts are marked explicitly as spam by the subreddit’s moderators. Furthermore, we treat post scores as a quality signal as this score captures up/down votes from Reddit users, and consider any post with a score of less than one as low-quality content. The lengths of the video files vary from a few seconds to multiple hours. We avoid long videos in our dataset because they can contain multiple events of different kinds and are very hard to process. We only keep videos that are longer than 2 seconds and shorter than 60 seconds. After applying our filters, our final dataset contains **26,954** video files from **1,873** different games.

### **Labelling videos with the game name**

In order to simulate the realistic scenario in which a game developer would search a repository of gameplay videos for a specific game, we extract the game name for each gameplay video from the title of its respective post. Detecting the game’s name from a GamePhysics submission is not straightforward. While there is a community guideline that suggests including the name of the game in the submission’s title, people often forget to include the game name or use several aliases for the game name, meaning the task of detecting the game name can be hard. For example, ‘GTA V’ is a widely-used alias that refers to the ‘Grand Theft Auto V’ game. To address this issue, we created a second custom crawler to search game name keywords in Google and subsequently map them to the full game name. Google search results provide a specific section called the Knowledge Panel that contains the game name, as well as other relevant game information such as initial release date, genre, development studio(s), and publisher.

### 3.5.2 Pre-processing the videos

As discussed in Section 3.4.2, our approach can search a large repository of gameplay videos more efficiently by pre-processing the embedding vectors of every frame for each video in the repository before inputting any text queries. Therefore, for our dataset to be suitable for our approach, we pre-process all videos in the `GamePhysics` dataset before proceeding with any experiments. We pre-processed all 26,954 videos using a machine with two NVIDIA Titan RTX graphics cards, but it is certainly possible to perform this step with less powerful graphics cards too. It is worth noting that this is by far the most computationally expensive step in our approach.

## 3.6 Preliminary evaluation setup

In this section, we describe the preliminary evaluation of our approach on the `GamePhysics` dataset through a diverse set of experiments. To evaluate our video search method, we conducted multiple experiments with varying difficulty levels. The main obstacle to evaluating our search system is the lack of a benchmark dataset. To this end, we designed two experiments with corresponding sets of queries to shed light on the capabilities of our proposed method.

### 3.6.1 Experiment overview

In two experiments, we evaluate the accuracy of our approach when retrieving videos with particular objects in them. The results for this step indicate the generalization capability of the model.

### 3.6.2 Selecting CLIP architectures

To understand the relative performance of the available ResNet-based and vision transformer-based CLIP models, we tried the RN101 and ViT-B/32 backbone architectures. We chose these backbones as fair baseline comparisons because they are the largest backbone architectures in their respective families, assuming we stipulate

Table 3.1: Selected games for the preliminary evaluation of our approach. All selected games are open-world.

Game	Key	Genre	Visual style	Reason for inclusion	Videos
Grand Theft Auto V	GTA	Action-adventure	Realism	Variety of vehicles	2,230
Red Dead Redemption 2	RDR	Action-adventure	Realism	Historical style	754
Just Cause 3	JC3	Action-adventure	Realism	Physical interactions	680
Fallout 4	F4	Action role-playing-game	Fantasy realism (Retro-futuristic)	Unique look and feel	614
Far Cry 5	FC5	First-person shooter	Realism	First-person camera	527
Cyberpunk 2077	C77	Action-adventure	Fantasy realism (Futuristic)	High-quality lighting	511
The Elder Scrolls V: Skyrim	ESV	Action role-playing-game	Fantasy realism	Magical effects	489
The Witcher 3: Wild Hunt	W3	Action role-playing-game	Fantasy realism	Mythical beasts	387

equivalent input image sizes ( $224 \times 224$ ). For comparison, the ViT-B/32 backbone architecture has 151 million total parameters, while the RN101 backbone architecture contains 119 million parameters. We selected the largest architectures as we only perform inference and no training.

### 3.6.3 Selecting video games

Our dataset contains videos from 1,873 different video games, and the differences in their high-level characteristics, such as genre, visual style, game mechanics, and camera view, can be vast. Therefore, we performed a comprehensive evaluation in both experiments with 8 popular video games that differ in their high-level characteristics. The only uniting characteristic of our selected games is that they have open-world mechanics because developers of open-world games would find particular benefits from an effective video search for bug reproduction. Open-world games allow a player to freely explore the game world, providing a very large set of potential interactions between the player and the game environment. Open-world games are therefore more likely to suffer from game physics bugs that are difficult to reproduce. Table 3.1 shows the selected games, as well as some game characteristics and the reason for inclusion. In total, 23% of videos in the GamePhysics dataset are from these 8 video games (6,192 videos).

### **3.6.4 Query formulation**

To come up with a set of relevant search queries in the experiments, we randomly picked 10 videos from each of the 8 selected games. The first author manually reviewed each of the 80 samples to understand what was happening and how we could describe events in gameplay videos that contain bugs. This sampling process helped us pick relevant objects and events to use in our queries.

### **3.6.5 Preliminary experiment 1: simple queries**

In this experiment, we searched for specific objects in videos, e.g. a car. Our main objective in this experiment is to demonstrate the capability of our system for effective zero-shot object identification. As a reminder, we never trained or fine-tuned our neural network model for any of these experiments or any video game. We created 22 distinct queries for Experiment 1, including transportation vehicles, animals, and special words describing the weather or environment. For this experiment, we wanted our approach to operate with very high detectability and to detect smaller variations in the video, and so we selected our first aggregation method, i.e. using maximum frame score per video (Section 3.4.3).

### **3.6.6 Preliminary experiment 2: compound queries**

Continuing our evaluation, we search for compound queries, i.e. queries in which an object is paired with some descriptor. Similar to Experiment 1, we only use compound queries that are relevant to each video game. For example, in the previous experiment, we searched for videos in the Grand Theft Auto V game that contained a car, but in this experiment, we evaluate the performance of our approach when searching for objects with a specific condition, like a car with a particular color. For this second experiment, we created a set of 22 compound queries and again selected our first aggregation method (using maximum frame score per video).

Table 3.2: Average Top- $k$  accuracy (%) per game for simple queries (Preliminary Experiment 1).

		GTA	RDR	JC3	F4	FC5	C77	ESV	W3
<b>ViT-B/32</b>	<b>Top-1</b>	74	71	61	65	50	55	54	54
	<b>Top-5</b>	89	86	67	71	88	73	62	62
<b>RN101</b>	<b>Top-1</b>	84	50	61	59	59	43	62	62
	<b>Top-5</b>	89	79	83	82	94	71	92	85

Table 3.3: Average Top- $k$  accuracy (%) per game for compound queries (Preliminary Experiment 2).

		GTA	RDR	JC3	F4	FC5	C77	ESV	W3
<b>ViT-B/32</b>	<b>Top-1</b>	68	88	56	43	31	50	56	56
	<b>Top-5</b>	100	100	81	64	69	75	89	67
<b>RN101</b>	<b>Top-1</b>	84	88	31	36	56	67	33	44
	<b>Top-5</b>	95	100	75	79	94	83	78	56

### 3.6.7 Evaluating the experiments

#### Evaluating preliminary experiments.

In the first and second experiments, we assess the detectability of our approach by measuring Top-1 and Top-5 accuracy. This is because for our approach to be useful to a game developer, the search system should be able to reliably identify objects specified in the text queries. Top- $k$  accuracy is a binary measure; if there is a correct result in the Top- $k$  results, the accuracy is 100%, otherwise, the accuracy is 0% – there are no possible values in between.

### 3.6.8 Results for preliminary experiments

In this section, we present experimental results to examine our proposed search system’s ability.

Table 3.4: Average Top- $k$  accuracy (%) per query for simple queries (Preliminary Experiment 1).  $N$  is the number of games searched.

Query	$N$	ViT-B/32		RN101	
		Top-1	Top-5	Top-1	Top-5
<i>Airplane</i>	4	75	100	100	100
<i>Bear</i>	5	80	100	60	100
<i>Bike</i>	4	50	75	50	100
<i>Bridge</i>	8	88	88	50	100
<i>Car</i>	5	80	100	80	100
<i>Carriage</i>	4	50	50	75	100
<i>Cat</i>	6	33	50	33	67
<i>Cow</i>	8	63	75	25	75
<i>Deer</i>	7	57	71	75	100
<i>Dog</i>	8	25	38	38	63
<i>Fire</i>	8	88	100	100	100
<i>Helicopter</i>	5	60	60	60	100
<i>Horse</i>	3	67	100	100	100
<i>Mountain</i>	7	100	100	100	100
<i>Parachute</i>	2	0	67	67	100
<i>Ship</i>	8	50	63	38	75
<i>Snow</i>	6	67	83	33	50
<i>Tank</i>	3	67	67	100	100
<i>Traffic Light</i>	5	40	40	20	20
<i>Train</i>	5	80	100	17	67
<i>Truck</i>	4	75	100	100	100
<i>Wolf</i>	6	17	50	86	86
<b>Average</b>	<b>5.5</b>	<b>60</b>	<b>76</b>	<b>64</b>	<b>86</b>

### Results for simple queries (preliminary experiment 1)

In the first experiment, we measured the Top-1 and Top-5 accuracy of our system with simple queries. The average accuracy for experiment 1 per game can be seen in Table 3.2, and per query in Table 3.4. The overall average Top-1 accuracy and average Top-5 accuracy of ViT-B/32 is 60% and 76% respectively, and for RN101 we have 64% and 86% respectively. These results show that our system can identify a majority of objects without fine-tuning or re-training.

### Results for compound queries (Preliminary Experiment 2)

In the second experiment, we measure the Top-1 and Top-5 accuracy of our approach with compound queries. The average accuracy in Experiment 2 per game can be seen in Table 3.3. For the second experiment, we find that our approach shows particularly high performance for all of our selected games, except for The Witcher 3: Wild Hunt. Our approach achieves an overall average Top-5 accuracy of **78%** using ViT-B/32 and **82%** using the RN101 model. These results show that our approach is flexible enough to effectively search gameplay videos with compound queries.

## 3.7 In-depth evaluation

Our preliminary findings demonstrate the effectiveness of two CLIP architectures in retrieving relevant gameplay video clips using simple queries. We expand our assessment by focusing on finding relevant videos using descriptions of bugs written in natural language. Each bug can be described using simple phrases that explain the issue. We use such descriptions as queries to search videos to find instances of that particular bug.

The in-depth evaluation reported in this section was done in collaboration with Ubisoft La Forge.

### 3.7.1 Experiment setup

In this experiment, we search videos with bug queries, i.e., phrases that describe an event in the game that is related to a bug. We create a new dataset in which each video is associated with a list of text descriptions detailing the bugs occurring within the video. It is important to note that these text descriptions are not utilized for retrieving videos, but rather serve as a means to evaluate the retrieval performance. For this particular experiment, we choose a game from the Assassin’s Creed<sup>2</sup> franchise.

We rank videos based on their similarity to a given bug query and return a list of videos sorted by rank. We report the Top- $k$  accuracy for various values of  $k$  ranging from 1 up to 50. The accuracy @25 and @50 is particularly relevant, as reviewing each result takes roughly one second. By presenting up to 50 results on a single page, users can efficiently review all the results in under a minute<sup>3</sup>. In order to enhance the efficiency of the review process, we provide a segment of the video that closely corresponds to the query and generate a GIF animation, which allows for a quick review of the results. This approach is especially advantageous for important bugs, as it allows for greater effort to be invested in identifying a relevant video.

To offer a comprehensive understanding of model performance and resource requirements, we present GPU memory usage for each model, in addition to the performance metrics. By examining the accuracy and computational costs, we provide a thorough analysis of the factors to consider when selecting a model for real-world applications.

### 3.7.2 Dataset and labeling videos

We selected all 278 videos from an Assassin’s Creed game from our `GamePhysics` dataset. We labeled all the videos manually and annotated them with natural language descriptions. We allowed multiple labels for each video since multiple bugs can occur in one video. We used an iterative process to label videos and in each itera-

---

<sup>2</sup><https://www.ubisoft.com/en-ca/game/assassins-creed>

<sup>3</sup>A sample search result is available at this URL.

tion, we ensured that different occurrences of the same bug have been labeled with a unique bug description. That is, if multiple videos exhibit the same bug but in different places, we assign an identical label to all videos. After filtering out low-quality videos (those with low resolution and significant degradation due to video encoding) and eliminating those without any apparent bug, our refined dataset ultimately contains 220 videos. Some sample bug descriptions can be seen in Table 3.5.

Table 3.5: Sample queries

#	Description	# Videos
1	An axe is floating in the air.	1
2	A horse is walking in the air.	1
3	A whale is floating in the air.	1
4	A person is shaking very fast.	2
5	A sword is flying in the air.	2
6	A floating boat is rotating very fast.	4
7	A person swimming in the air.	4
8	A boat is floating in the air.	6
9	A person is standing in the air.	7
10	A horse is running on its two legs.	8
11	A person is falling from the sky.	12

### 3.7.3 Models

In total, we evaluated five architectures for CLIP, and two for OpenCLIP [71], an open-source replication of CLIP. In addition, we evaluated four video-text contrastive models (XCLIP [134], BridgeFormer [49], CLIP4Clip [119] and Frozen-In-Time [12]), which work on more than one frame in contrast to CLIP, which operates at the individual frame level. Some gameplay bugs manifest themselves temporally, such as when an object shakes rapidly in the game. Video-text models can incorporate

temporal information and, in theory, have the ability to detect such bugs.

## **CLIP**

We studied five architectures for the original CLIP model, with different backbone architectures (ResNet and ViT). In particular, we include ViT-B/16, ViT-B/32, ViT-L/14, RN50x64, and ViT-L/14@336px in our study, which differ in architecture and parameter sizes. The input dimension of all the models is  $224 \times 224$ , with the exception of ViT-L/14@336px and RN50x64 which require an input image with the resolution of  $336 \times 336$  pixels and  $448 \times 448$  pixels, respectively.

## **OpenCLIP**

OpenCLIP is an open-source replication of CLIP that contains a diverse array of models with different sizes and architectures. We selected two of the largest and most capable OpenCLIP models; ViT-g/14 and ViT-H/14, to compare with the original CLIP. OpenCLIP models were trained on the LAION [164] dataset, consisting of 5.85 billion CLIP-filtered image-text pairs. Both of the selected models have an input spatial dimension of  $224 \times 224$  pixels.

## **XCLIP**

XCLIP consists of a transformer [191] model trained on features extracted by the original CLIP model to incorporate and fuse temporal signals and information. Essentially, this model receives a sequence of 32 frames with a resolution of  $224 \times 224$ , as its input and then compares the similarity of this sequence to a given text.

## **CLIP4Clip**

CLIP4Clip is a straightforward expansion of the original CLIP model for learning representations of video-text data. CLIP4Clip first extracts CLIP embeddings for each frame and then combines them to create an embedding for the entire video. Three types of aggregation methods can be used with CLIP4Clip: ‘parameter-free’,

‘sequential type’, and ‘tight type’. The first method is a basic averaging technique that does not require any training. We opt for this aggregation method because it is not limited to a particular domain, and more importantly, it delivers similar and competitive results across various benchmarks [119]. The CLIP4Clip model employs a ViT-B/32 model with an input resolution of  $224 \times 224$  pixels. Note that CLIP4Clip is very similar to the approach used in our preliminary evaluation (the approaches were developed and published in parallel).

### **Frozen-In-Time**

Frozen-In-Time is a transformer-based model that effectively learns a joint representation of video and text. The model supports single and multi-frame inputs, allowing it to leverage the strengths of both image and video datasets. This model treats image inputs as a special case of videos that are “frozen” in time. This model accepts frames and images with a  $224 \times 224$  pixels resolution.

### **BridgeFormer**

BridgeFormer is a video-text pre-training model based on a Multiple Choice Questions (MCQ) formulation. This model is trained to answer questions constructed by the text features, based on video features. This formulation enables the model to capture additional regional content and temporal dynamics in videos, in addition to semantic associations between local video-text features. BridgeFormer works on the input resolution of  $224 \times 224$  pixels.

### **Random baseline**

Our labeled dataset comprises only 220 videos, which may result in inflated accuracy rates across all models. We include a **Random Baseline** to evaluate the efficacy of our proposed methods compared to randomly reviewing videos. This model assigns random similarity scores to query and video pairs, returning a random order of videos

for queries. Therefore, this random retriever provides a way to determine the actual benefit of the proposed models.

### Implementation details

We used official libraries for the CLIP<sup>4</sup> and OpenCLIP<sup>5</sup> models. Since CLIP and OpenCLIP work at the individual frame level, we used two aggregation methods for all models of this kind, *max* and *mean*. For a given query and a video, we calculate the similarity score between all embeddings of frames and the embedding of the given query. For the *max* aggregator, we choose the maximum score among all frames with the query and use it as the final score, and for the *mean* aggregator, we use the average of the top 5 similar frame count scores (as discussed in Section 3.4).

We used the XCLIP implementation from the Hugging Face library [202]. Since XCLIP expects an array of 32 frames as input, we fed the model a rolling window of 32 frames and stored multiple embeddings per video. For a given query and a video, we calculated the text embedding of the video with all window embeddings and used the *max* aggregator to calculate the final similarity score.

We employed the Towhee framework<sup>6</sup> to implement the CLIP4Clip, Frozen-In-Time, and BridgeFormer models. For each input video, we uniformly sampled frames and fed them to the respective model. Specifically, we utilized the *uniform temporal subsampling*<sup>7</sup> technique to extract frames from each video.

### 3.7.4 Experimental results

Table 3.6 shows the results of our experiment, and Figure 3.4 illustrates samples of successful retrieval. The considerable gap between the results of the **Random Baseline** and the other models demonstrates that the selected models are genuinely detecting relevant information and are not simply benefiting from chance.

---

<sup>4</sup>Github - OpenAI CLIP

<sup>5</sup>Github - mlfoundations OpenCLIP

<sup>6</sup><https://towhee.io/>

<sup>7</sup>PyTorch Video Documentation

**ViT-L/14 and ViT-L/14@336px perform best, with ViT-L/14@336px marginally better.** Our results show the ViT-L/14@336px model consistently outperforms all other tested models. It achieves a Top-10 accuracy of 49.44% and its Top-50 accuracy reaches up to 82.58%, suggesting it can successfully identify relevant videos for most queries.

**Both input resolution and architecture are important factors in bug video identification performance.** Among all tested models, some have different input resolutions, e.g., RN50x64 has an input size of  $448 \times 448$  pixels, and ViT-L/14@336px has  $336 \times 336$  pixels. Higher-resolution inputs provide more information for the model to process, paving the way for better performance at the expense of increased computational resources (discussed in Section 3.7.5). Comparing performance among architectures based on ViT (ViT-L/14 vs ViT-L/14@336px), shows indeed that higher resolution inputs lead to better performance. However, the ViT architecture performs better than the ResNet architecture (RN50x64 vs ViT-L/14@336px), despite the fact that ResNet uses a higher input resolution. This suggests that both model size and architecture play roles in the final performance of a model.

**Video-text models consistently perform worse than image-text models for gameplay videos.** Among all tested video-text models, XCLIP achieves better Top-1 and Top-5 accuracies. However, as we gradually increase the accuracy threshold, the advantage of XCLIP seems to diminish, and all video-text models perform similarly and consistently worse than image-text models. This low performance at first is counter-intuitive since video-text models have the ability to exploit temporal information in videos and should perform better than image-text models. However, this low performance happens because these models have been fine-tuned on real-world videos, which are very different from gameplay videos, i.e., gameplay videos are considered out-of-distribution for these models. This finding is aligned with the observations reported by other previous studies [203, 84].

**Image-text models cannot exploit the temporal dimension but are able to use other information to find relevant videos.** A subset of bugs in our dataset manifest themselves over time, and one can not detect them from a single frame. For instance, when an object is unintentionally *shaking*, individual frames do not reflect this behavior and we need to incorporate the temporal aspects of the video. Even though both CLIP and OpenCLIP are image-text models and lack an understanding of the temporal aspects of a video, they consistently performed better than video-text models. After carefully analyzing some queries, we found that image-text models oftentimes utilize a portion of the query to retrieve the most relevant videos that represent the query as closely as possible. For instance, for the query “*A person is being teleported into the sky*”, CLIP retrieves videos showing aerial views of the game world, which turn out to be the correct videos for the given query.

### 3.7.5 GPU memory usage and search speed analysis

The ViT-L/14@336px model outperforms its counterparts at every accuracy threshold; however, this performance comes with the drawback of higher GPU memory usage, which might hinder its usage compared to other models. In this section, we present the GPU memory requirements for each model. We used the `gpustat`<sup>8</sup> Python package to determine GPU memory usage for each model. We created a sample script that loads and passes input to the model and measures the maximum memory utilization. For both CLIP and OpenCLIP families, we use a batch size of 256 images as input. This batch size is roughly equivalent to the length of all frames in a short video (e.g., a 10-second video with 24 frames per second). For all video-text models, we use an input that matches the network input dimensions. These measurements provide an approximate estimate of the GPU memory required to calculate the similarity of a gameplay video with a text query. Table 3.7 shows the GPU memory usage measurements. As we can see, the OpenCLIP ViT-g/14 model requires the most

---

<sup>8</sup><https://github.com/wookayin/gpustat>

Table 3.6: The performance of the studied models on our labeled dataset of 220 videos – The bold numbers highlight the best-performing model. (%)

	Model Name	Modality	Aggregator	Top-1	Top-5	Top-10	Top-15	Top-20	Top-25	Top-50
	Random Baseline	video-text	-	3.93	6.74	8.99	12.92	16.29	20.22	32.02
CLIP	ViT-L/14	image-text	Max	17.42	33.15	42.70	52.25	55.62	62.36	79.21
	ViT-L/14@336px	image-text	Max	21.91	38.20	47.19	52.81	56.74	62.92	80.34
	ViT-B/16	image-text	Max	15.73	31.46	41.57	48.31	51.12	53.37	75.28
	ViT-B/32	image-text	Max	15.73	25.84	34.83	42.70	48.31	53.93	68.54
	RN50x64	image-text	Max	16.29	35.39	41.57	48.31	51.69	56.18	74.16
	ViT-L/14	image-text	Mean	19.66	34.27	44.94	55.06	58.99	62.92	82.02
	ViT-L/14@336px	image-text	Mean	<b>22.47</b>	<b>40.45</b>	<b>49.44</b>	<b>56.74</b>	<b>62.36</b>	<b>66.85</b>	<b>82.58</b>
	ViT-B/16	image-text	Mean	20.22	32.58	41.01	46.07	53.37	57.30	79.21
	ViT-B/32	image-text	Mean	15.73	32.02	41.57	46.07	53.37	55.62	73.03
	RN50x64	image-text	Mean	16.85	35.39	42.70	50.56	55.06	61.24	78.65
	OpenCLIP	ViT-H/14	image-text	Max	16.85	29.21	36.52	47.75	51.69	55.62
ViT-g/14		image-text	Max	17.42	27.53	35.96	43.82	50.00	52.81	69.66
ViT-H/14		image-text	Mean	19.10	32.02	42.13	48.88	52.25	55.06	73.60
ViT-g/14		image-text	Mean	16.85	31.46	38.76	47.75	53.37	59.55	73.60
	XCLIP [134]	video-text	Max	10.11	21.91	29.78	37.64	43.26	51.69	69.66
	CLIP4Clip [119]	video-text	-	7.87	21.35	33.71	41.57	47.19	52.25	66.85
	BridgeFormer [49]	video-text	-	7.87	18.54	29.78	40.45	46.63	52.81	70.22
	Frozen-In-Time [12]	video-text	-	6.18	17.98	28.09	33.71	39.89	47.75	66.29

memory, but according to our results in Table 3.6, its performance falls behind the CLIP ViT-L/14@336px model. Also, the ResNet model requires considerably more GPU memory than the better-performing ViT models. Hence, model performance is not necessarily correlated with GPU memory requirements.

Table 3.7: GPU memory usage of the studied models.

Model	VRAM (MB)	Input Size	Embedding Size
ViT-L/14	5,861	224 × 224px	768
ViT-L/14@336px	11,803	336 × 336px	768
ViT-B/16	4,361	224 × 224px	512
ViT-B/32	2,551	224 × 224px	512
RN50x64	21,829	448 × 448px	768
ViT-H/14	15,559	224 × 224px	1024
ViT-g/14	13,275	224 × 224px	1024
XCLIP	7,979	224 × 224px	512
CLIP4Clip	2,987	224 × 224px	512
BridgeFormer	3,463	224 × 224px	512
Frozen-In-Time	3,405	224 × 224px	512

An additional factor that impacts the practical application of our method is the search speed, which refers to how fast a video can be retrieved. Our method requires a one-time pre-processing of videos into embeddings, which can conveniently be performed overnight or as part of a company’s continuous build process. This enables subsequent search queries to be efficiently addressed using the generated embedding files. By employing the Faiss library [75], the search process can be executed in a matter of milliseconds, even when handling a dataset comprising millions of frames. This efficiency makes the search both practical and realistic in real-world circumstances.

### 3.8 Discussion and limitations

In this section, we discuss the strengths and weaknesses of our approach, based on the results of our experiments. Figure 3.4 shows several example video frames from



(a) Video of ‘A faceless person.’



(b) Video of ‘A horse is running on its two legs.’



(c) Video of ‘A boat is floating in the air.’



(d) Video of ‘A person is stuck in a tree.’

Figure 3.4: Relevant gameplay videos identified using our approach with bug queries in a game from the Assassin’s Creed franchise. Our method can effectively retrieve highly specific bugs using textual descriptions, without depending on specialized vocabulary or technical terms.

videos identified when searching gameplay videos with text queries using our approach. These examples help to illustrate the promising potential of our approach. Given that our method does not require any training in gameplay videos, our zero-shot setup for detecting objects in videos is promising. During our experiments, the first author manually analyzed each video returned by our search approach, including false positives. Below, we highlight our observations about retrieving buggy gameplay videos using our approach:

### Handling technical and non-technical terms

Our observations suggest CLIP understands these technical terms, and sometimes, describing a bug in simple words retrieves relevant videos. For instance, Figure 3.4(a) illustrates a game character without facial features. CLIP can retrieve this video

using both a specialized term like “*a low-poly face*,” and a common and non-technical description such as “*a faceless person*.” It is worth noting that non-technical users and gamers could describe bugs using non-technical terms in their bug reports. For example, game reviews tend to contain implicit bug reports which are often written by non-technical users [103]. Our approach would help developers to gather more information about such bug reports.

### **Adversarial poses**

An important category of problems is the unusual pose of familiar objects. As extensively tested and reported by Alcorn et al. [5], neural networks occasionally misclassify objects when they have different poses than what they used to have in the training set. For example, consider a neural network that can detect a “*car*” in an image. It is possible to find a particular camera angle for which the neural network can not detect the “*car*” in that image. In a dataset of natural images, there may be many examples of cars, but the camera angle and the position of the car relative to the camera usually does not vary much. A neural network trained on such data will struggle to detect a car when it sees it from a very unusual angle (e.g. when it is positioned vertically).

### **Embedding size**

In our experiment, we used a range of models with different architectures and modalities. A key variable in these models is the embedding size. The embedding size is a fixed-length vector that summarizes the contents of an image or video. For the models we tested, the embedding size varied from 512 to 1024. A natural question arises: *Are larger embedding sizes necessarily better?* Our findings indicate that there is no direct correlation between embedding size and performance. This is evident by comparing the models like ViT-L/14@336px and ViT-g/14 which have dimension sizes of 768 and 1024 respectively, but the performance of ViT-L/14@336px is consistently

better than ViT-g/14. One explanation is that the final embedding size alone does not guarantee the quality of the embedding, and this factor is largely dependent on the architecture and the computational power spent on the training rather than other factors.

### **Rapid camera motion causes confusion in video-text models**

Among video-text models, especially XCLIP, we noticed an unexpected behavior of the model when the video contains rapid camera motion. After reviewing a subset of queries that result in retrieving unrelated videos, we observed that the top matching video often contained rapid camera movements, such as orbital movement (which is sometimes known as the “death cam”<sup>9</sup> in gaming communities). These movements, which are present in many gameplay videos, can be considered ‘adversarial samples,’ create distractions and leading to incorrect query-video matching.

### **Video quality**

Our dataset is constructed from internet videos, which are often in low-resolution formats. However, due to lossy and heavy video compression, a significant predictive signal is lost, which can pose a challenge for any vision model. Furthermore, we have observed that sometimes, the extracted frames have very low quality. While human vision has no problem detecting objects in the video, sometimes CLIP models cannot properly detect them. It is worth mentioning that, in a controlled testing environment within a gaming company, high-quality videos can be recorded without incurring significant costs, thus avoiding any problems that may arise due to video quality.

### **CLIP can recognize texts**

Another observation is about text patches inside gameplay videos. CLIP can *read* the text inside an image as well. This feature is not something that the model was explic-

---

<sup>9</sup>You can view a sample by watching [🔗](#) this video.

itly trained for, but rather some emergent behavior of pre-training in a contrastive setting. Sometimes searching a particular text query will result in retrieving a video that ignores the meaning of the text query, but the image contains that text. For example, if any video frames include a text field containing “*a blue car*”, searching for the query “*a blue car*”, will retrieve that video. Obviously, depending on the use case, this can be treated as both a feature and a bug.

### **Confusing textures and patterns in the images**

The textures and patterns can pose influential distractions and confusion for the neural network model in our approach. Sometimes a part of a game environment has a texture similar to another object. For example, our model confuses a striped colored wall in the Grand Theft Auto V game with a “*parachute*.” This category of problems is hard to encounter globally because each game has a diverse look and feel and creative artistic directions.

### **Confusions in accurately recognizing objects with CLIP**

The CLIP model struggles to accurately distinguish between various objects, with vehicles and animals serving as notable examples. In our analysis, we discovered instances where the search results partially matched the textual description but mistakenly featured a different object from the intended category. Our manual evaluation showed that CLIP often make errors in identifying various vehicles, such as cars, airplanes and tanks. Likewise, the model struggles to differentiate between four-legged animals, including dogs, cats, wolves, cows, horses, etc.

## **3.9 Threats to validity**

In this section, we discuss potential internal and external factors that may affect our findings.

### 3.9.1 Threats to internal validity

#### Size of dataset in the industry-related evaluation

The validation set used in the industry experiment comprises 220 videos. Although this dataset encompasses all videos from our selected game, it remains relatively small compared to the entire `GamePhysics` dataset. Labeling a dataset is a laborious, time-consuming, and costly task. In this study, we selected a single, representative game illustrating the challenges of an underdeveloped industry game. When adapting our method, it is important to consider that depending on video game visuals and the size of the dataset, the number of false positives can increase or decrease.

#### Effect of subsampling on the performance

Our straightforward extension of image-text models for video data entails initially encoding all frames using the model, followed by an aggregation step, to select one or several frames that exhibit the highest degree of alignment with the given query. Conversely, video-text models are designed to inherently accept a collection of frames as input and subsequently generate an embedding. Yet, these models do not work on all frames, and in the initial step, they subsample the input video. We employed *uniform sampling* to select specific frames for these models; however, this approach may result in the loss of frames, potentially leading to diminished performance.

### 3.9.2 Threats to external validity

#### Dataset generalizability and video length

As our dataset predominantly consists of gameplay videos that contain game physics bugs, our approach may not be as effective for other datasets of gameplay videos that contain other types of bugs. Non-curated datasets may contain many more videos of non-buggy gameplay, for example, if using gameplay streaming footage. Additionally, we excluded long (>60 seconds) videos, meaning our approach may not be effective for long videos. We also ignored videos with scores of zero from the `GamePhysics`

subreddit. After manually checking a random sample of low-scored posts we observed that a score of 0 almost always indicated low quality/span/etc. This threshold might not be applicable for other subreddits. Future research is required to evaluate our approach with long videos and non-curated datasets.

### 3.10 Conclusion

In this chapter, we proposed a novel approach to mine large repositories of gameplay videos by leveraging the zero-shot transfer capabilities of CLIP to connect video frames with an English text query. Our method can find objects in a large video dataset using simple and compound queries. Additionally, our approach demonstrates promising performance in detecting specific (bug-related) events, suggesting potential applicability in automatic bug identification for video games. Without fine-tuning or re-training for the video game domain, our method shows surprising effectiveness in most studied games.

In a preliminary study, we evaluated our method on a dataset of **6,192** videos from 8 games with different visual styles and elements to assess its capability in identifying objects in a gameplay video date. Our experiment confirms that CLIP can generalize to video game data and can effectively identify objects with simple and compound queries.

In a controlled setting, we evaluated our method on a video game from the Assassin’s Creed franchise to understand the benefits and limitations of our method in a real game development environment. Our findings showcase the usefulness and potential of our technique in retrieving gameplay videos using natural language descriptions of bugs. The Top-10 accuracy of our best-performing model reaches 49.44. Given out-of-distribution nature of video game footage and video game bugs, this result is quite promising. Furthermore, our research demonstrates that, by employing our CLIP-based search method, it is possible to retrieve relevant videos for up to **82.58%** of queries. Despite potentially needing to review up to 8 minutes of video footage per

query, this approach highlights the practicality of our method in aiding developers and quality assurance teams to rapidly identify and resolve issues within their games.

Additionally, we carried out a comparative study between various video-text and image-text models to assess their performance. Initially, it was anticipated that video-text models would be at an advantage, given their ability to integrate temporal information. However, our findings demonstrate that all tested video-text models underperformed in comparison to their image-text counterparts. This discrepancy can be attributed to factors such as out-of-distribution visuals of the game as well as camera motions that are very different from those in regular video datasets.

In conclusion, our novel method for retrieving gameplay videos demonstrates a significant advancement in the field of video game quality assurance. Our method provides a rapid and efficient approach to search through large quantities of video content, enabling quality assurance teams to rapidly identify relevant data related to game bugs. Finally, our approach lays the foundation for utilizing contrastive learning models for zero-shot bug identification in video games. Future work in this line of research will provide more insights into video game bugs and will pave the way to creating a new paradigm of automated bug detection methods for video games.

## Chapter 4

# GlitchBench: Can large multimodal models detect video game glitches?

**Bibliographic information:** This chapter was published as: Taesiri, Mohammad Reza, et al. “GlitchBench: Can large multimodal models detect video game glitches?” in the *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) 2024* [174].

Large multimodal models (LMMs) have evolved from large language models (LLMs) to integrate multiple input modalities, such as visual inputs. This integration augments the capacity of LLMs for tasks requiring visual comprehension and reasoning. However, the extent and limitations of their enhanced abilities are not fully understood, especially when it comes to real-world tasks. To address this gap, we introduce **GlitchBench**, a novel benchmark derived from video-game quality assurance tasks, to test and evaluate the reasoning capabilities of LMMs. Our benchmark is curated from a variety of unusual and glitched scenarios from video games and aims to challenge both the visual and linguistic reasoning powers of LMMs in detecting and interpreting out-of-the-ordinary events. Our evaluation shows that **GlitchBench** presents a new, interesting challenge to state-of-the-art LMMs. Code and data are available at: <https://glitchbench.github.io/>

## 4.1 Introduction

The video game industry boasts an estimated annual revenue of USD 217 billion [158] with a total of 3.2 billion gamers worldwide in 2022 [157]. Automatically detecting in-game glitches is, therefore, a highly demanding task, but that remains a long-standing challenge [154, 176, 175, 105, 229, 155, 27, 201, 133]. A *glitch* is an unexpected frame that occurs within a game due to either an unforeseen software bug, player actions, or unanticipated interactions between game elements and does *not* result in a program crash. From a computer vision perspective, glitch detection involves recognizing an extremely wide spectrum of long-tail video frames, from rendering (*e.g.*, 3D objects with missing textures; Fig. 4.2f), unrealistic physics (*e.g.*, two people sitting in an invisible car; Fig. 4.2b), to semantic errors (*e.g.*, indoor rain; Fig. 4.1). Unlike software bugs that can be caught by examining the code alone, glitches are more non-trivial to detect because they are mostly the result of a one-time event that unexpectedly happens in-game.

A holy grail of game quality assurance is to build a general glitch detector that works for any game of any genre and mechanics. We set the first step toward this goal by building `GlitchBench`, an evaluation benchmark of 593 glitches, leveraging the public’s crowd knowledge from the game community’s reports on [reddit.com/r/GamePhysics](https://www.reddit.com/r/GamePhysics). The glitches span across 205 games of various genres. Each glitch has a video clip, a representative frame, a one-line description, and a reference to a corresponding Reddit thread where gamers discussed the error.

Large image-text, multimodal models (LMMs), such as GPT-4V [138], are improving at an unprecedentedly fast pace. They excel in many existing tasks, including object detection [209, 113], multi-step reasoning [25, 94, 4, 10], and detailed image captioning [138, 104, 108, 210, 142]. Testing LMMs on `GlitchBench` may yield important findings not only to the game industry but also to the Artificial Intelligence (AI) community because glitch detection requires a combination of knowledge and under-

GPT-4V fails to focus on the rain inside the room



**Question:** What is wrong with this image?

**Ground truth:** It is raining inside the building

**GPT-4V:** There's nothing inherently "wrong" with this image; however, if you're referring to inconsistencies or peculiarities that might not align with the established Batman canon, the red Batman suit is unusual since Batman is traditionally known in black or very dark colors. The use of red could be an artistic choice for the game, an alternate suit available within the game's universe, or indicative of a different character who takes on the Batman mantle.

Figure 4.1: The image depicts a screenshot in which it rains inside a room. While the rain should be what is wrong with the image, GPT-4V fails to reason correctly and instead focuses on the color of Batman's costume. Note that the ground truth is never presented as part of the prompt in our study.

standing of image aesthetics, computer graphics, physics and commonsense reasoning (skills that are often tested individually in a benchmark [20]).

In this chapter, we evaluate how well LMMs perform in detecting glitches from a single frame. Our main findings and contributions include:

1. We introduce `GlitchBench`, which contains 330 glitch-free and 593 glitch screens taken from 205 games for evaluating LMMs (Sec. 4.3).
2. We evaluate 11 state-of-the-art LMMs, including GPT-4V [138] and LLaVA [108] on our benchmark and in comparison with the performance on 6 other common benchmarks (Sec. 4.4).
3. LMMs are better at detecting glitches that violate simple physical laws (*e.g.*, a car flying in the air) than other more subtle glitches (*e.g.*, human limbs in an implausible pose; Fig. 4.6).
4. The state-of-the-art model on `GlitchBench` is GPT-4V with 43.4% accuracy. In the extensive captioning setup, we estimated the upper limits of models, and GPT-4V can achieve an accuracy of 64.9%, which is almost twice that of LLaVA, the second-best model (30.5%).
5. In sum, there exists a headroom of 30–35% on `GlitchBench` for future LMM models to improve, presenting an interesting challenge to the AI community.

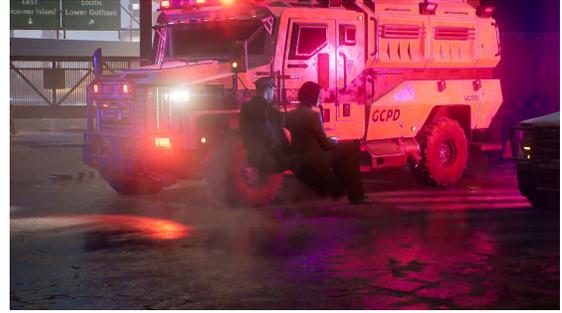
## 4.2 Related work

### 4.2.1 Multimodal, image-text datasets

Recently, there has been rapid development of large multimodal models that can process multiple modalities, including visual and textual inputs. Existing datasets that come with human-generated image captions, such as COCO Caption [29], Nocaps [3], CapFilt: [97] and Flickr30k [145], can serve as a simple way to evaluate language



(a) A person stuck in a piece of furniture



(b) Two people driving an invisible car



(c) A rifle floating in the air



(d) A person is floating in the air



(e) The gun in the hand is missing



(f) The table cover has a placeholder texture

Figure 4.2: Sample images from the `GLitchBench` showing glitches in various games with distinct styles. Samples (a)–(e) are captured from online videos, while sample (f) is generated inside the Unity game engine.

models. By providing the image, we can ask a model to describe it and then compare the generated caption with the ground truth [108, 211, 111]. Image captioning is a narrow domain and can be extended into visual question answering (VQA) by asking questions related to an image. Datasets like GQA [70], OK-VQA [126], VQAv2 [54], and Vizwiz [57] contain image-question pairs to probe the visual reasoning and understanding of LMMs.

Building upon simple VQAs, several benchmarks aim to increase the complexity of tasks over different dimensions. TextVQA [170], OCR-VQA [131] and TextCap[169] propose questions about the text shown in the image. ScienceQA [116] and Math-Vista [115] focus on scientific topics and charts, while VCR [216] and Sherlock [216] focus on commonsense reasoning. Moreover, AI2D [65] is directed at questions concerning scientific diagrams, and IconQA [117] targets the comprehension of abstract diagrams. Each of these benchmarks is designed to push the boundaries of VQA systems by introducing specialized content that requires advanced reasoning and understanding.

There are also comprehensive evaluation frameworks that assess multimodal language models across a wider spectrum of capabilities. These evaluations extend beyond visual and textual reasoning to encompass a variety of skills such as generation, question answering, adherence to instructions, and the application of commonsense logic. Notable among these are SEED-Bench [93] , MME [46], MMBench [114], MM-Vet [213], VisIT-Bench [20], which collectively serve to provide a robust measure of a model’s proficiency in handling tasks that integrate multiple modalities.

Unlike traditional datasets that contain queries about elements present in the image, our approach is novel in directing models to discern the atypical aspects, *i.e.*, glitches, with no linguistic hints provided. We show an image to the model and ask it to report unusual aspects of it. Such questions require a more integrated approach to visual and linguistic processing within an LMM to formulate a response.

### 4.2.2 Vision-language stress testing

Out-of-distribution (OOD) datasets have become a cornerstone for evaluating the capabilities and progress of machine learning models. In standard image classification, in particular the ImageNet [162] dataset, the introduction of datasets [62, 62, 64, 179] has underscored the importance of robustness and generalization in model evaluation. As we move from simple image classification tasks to more complex multimodal tasks,

there is an increasing need for similar OOD datasets that can comprehensively test the generalization abilities of LMMs.

There are several studies that stress test various aspects of vision from different angles, such as compositional and spatial reasoning [78, 50, 184, 73], objects placed out of context and implausible scenes [31, 56, 230], and the exploitation of language and vision priors [106, 45].

The closest benchmark to ours is Whoops [56], which is designed to challenge commonsense knowledge and reasoning in LMMs. However, our dataset differs in several ways: (1) The tasks in `GLITCHBENCH` come from real-world tasks, specifically video game quality assurance, and are not artificially created to test models. (2) Whoops requires cultural and background knowledge to answer; for example, *A panda bear is catching salmon fish* is unusual since pandas subsist almost entirely on bamboo. In contrast, our dataset contains samples that contradict basic commonsense and the physics of the world. (3) Finally, images in Whoops are synthesized using image-to-text models; they are clear without artifacts, centered in the image, and do not stress the visual side of the image, focusing on the context. In contrast, for `GLITCHBENCH`, models need to fully scan the image to identify its unusual aspects (Fig. 4.2), and there are many distracting elements present in the image, challenging them to focus on the correct part of the image.

### 4.2.3 Empirical analysis of recent LMMs

With the release of recent proprietary LLMs, such as GPT-4V and Bard [52], some studies attempt to evaluate and report the performance of these models on various benchmarks and tasks [204, 34, 150]. The main goal of these studies is to provide a comprehensive evaluation of the models across various well-established tasks and some narrow domains [200, 208]. The main difference between our work and these studies is that we propose a general, stress-testing benchmark to measure the generalization power of various LLMs, both proprietary and open source, on a specific,

glitch-detection task in the game industry.

## 4.3 GlitchBench

In this section, we describe the creation process of `GlitchBench`, a benchmark aimed at stress-testing visual perception and commonsense reasoning in LMMs, motivated by real-world game quality assurance tasks.

During development, video games go through many stages of testing to reach certain quality standards before release. However, even after release, they can still exhibit unusual in-game events, or glitches. Glitches, often viewed as annoying bugs, can also possess a humorous and entertaining aspect. Players frequently report glitches across various social media platforms, particularly on Reddit and YouTube. A critical aspect of understanding glitches is the requirement of commonsense knowledge about the basic laws of physics of the game’s universe, making them a suitable and practical candidate for testing machine learning models. Fig. 4.2 shows six samples from `GlitchBench`.

### 4.3.1 Constructing the dataset

`GlitchBench` contains two parts: (1) 513 samples shared by players of video games, *i.e.*, frames collected from online sources, and (2) 75 synthetic samples.

**Samples shared by players of video games:** To construct our dataset, we sampled 1,000 videos from the GamePhysics [176] dataset. This dataset consists of videos from a subreddit with the same name, containing gameplay video clips with unusual events and glitches.

Next, we conducted a manual review process to filter videos based on two criteria: (1) the presence of a glitch in the video, and (2) the potential for humans to detect the glitch from a single frame. The second criterion is key because certain glitches, such as those involving rapid shaking or changes in size over time, cannot be detected

from a still image alone.

After applying these filters, we extracted one frame from each remaining video, resulting in a collection of 650 samples. Our final round of manual reviews revealed two potential issues: (1) some glitches are not detectable from the extracted image and require more context to understand, and (2) some images contain the faces of gamers who streamed the content on an online platform (which could cause the LMM to identify these faces as what is wrong with the images). After removing videos that contain one of these issues, our final glitch set contains 513 images.

**Generating synthetic samples with Unity:** To enhance our dataset, we supplemented samples from the GamePhysics dataset with 75 synthetic examples created inside the Unity game engine. These samples were specifically designed to mimic a subset of common development-stage bugs [154, 175, 105] that are not readily available in online social media platforms and, hence, to diminish the survivor bias effect. These flaws are often fixed before the public release of a game through the quality assurance process of a game development company and are therefore not often posted on social media.

Our synthetic sample generation process involves the injection of three categories of glitches into each scene: (1) placeholder textures, (2) object mesh distortions, and (3) low-resolution textures.

**Glitch-free images:** Our focus is on glitch frames, as they are more challenging to capture and collect. However, to establish a baseline for comparison, we also included a set of glitch-free images. To accomplish this, we randomly selected gameplay walkthroughs from various games on YouTube. From these walkthroughs, we extracted a random subset of frames, resulting in the compilation of a dataset consisting of 330 frames sourced from a diverse array of games. The groundtruth captions for these glitch-free images is *“There is nothing wrong with this image”*.

### 4.3.2 Labeling the dataset

For all images, we provide a short description of the glitch present in the image. Our goal is to label the images briefly, highlighting only the unusual elements in simple language. For instance, if an image depicts a character with a contorted physique, the label would simply state, “*character has an unnatural body position*”.

It is important to highlight that some images can be described in many different ways. Diverse phrases such as “*falling from the sky*”, “*suspended in mid-air*”, or “*jumping in the air*” might all refer to a single event. Instead of handling such cases in the labeling process, in the evaluation process, we incorporate a language model to diminish the effect of this (see Sec. 4.4.1).

### 4.3.3 Categorizing the glitch types in the images

In this section, we provide a high-level categorization of glitches in our dataset. While there have been some attempts to provide a taxonomy of video game bugs [90, 188], these taxonomies do not provide descriptions that are adequate to automate bug categorization.

We propose a novel human-AI team-based method to build a categorization based on the descriptions of the images. This process is a collaborative effort between GPT-4 and humans, where GPT-4 suggests initial categories, and then humans refine these suggestions by providing feedback or asking the model to re-evaluate its output, harnessing the reflective ability of GPT-4 [168]. Finally, we manually bridge the resulting categories to those proposed by Lewis et al. [90] based on the semantics and instances of the glitches in our dataset.

**Process:** We prompt GPT-4 with all the glitch descriptions in our dataset and ask it to generate a categorization based on the descriptions and semantics of the glitches. In each subsequent iteration, we provide feedback in one of two ways: (1) we ask GPT-4 to review its previous answer through reflection, or (2) we explicitly instruct

the model to merge two categories that are semantically similar. We stop when the model no longer changes its answer through reflection or when we can no longer merge categories.

In the last step, to assign each image to a category, we prompt GPT-4 with the description of the glitch and the final categories and ask it to assign each image to one of them. The final categories, the number of instances, examples for each category, and the parent category proposed by Lewis et al. [90] are outlined in Table 4.1.

Table 4.1: Categorization of video game glitches in **GlitchBench**. Numbers highlighted in   show the number of images in each category. Categories highlighted in   show the corresponding categories proposed by Lewis et al. [90].

<b>Physics, Collision, and Spawn</b>	Images: 422
(Non-Temporal → Invalid position)	
<ol style="list-style-type: none"> <li>1. Objects and characters floating or stuck in the air (Fig. 4.2d).</li> <li>2. Characters or objects clipping through solid objects like walls, floors, or ground.</li> <li>3. Vehicles or characters falling under the game map.</li> </ol>	
<b>Animation and Pose</b>	Images: 75
(Non-Temporal → Invalid graphical representation)	
<ol style="list-style-type: none"> <li>1. Unusual or impossible body poses and positions (Fig. 4.6).</li> <li>2. Characters in a T-pose or with distorted body parts.</li> <li>3. Incorrect animations for certain actions.</li> </ol>	
<b>Rendering and Texture</b>	Images: 67
(Non-Temporal → Invalid graphical representation)	
<ol style="list-style-type: none"> <li>1. Mesh stretches or objects with distorted shapes.</li> <li>2. Missing textures or objects displaying a “default” placeholder texture (Fig. 4.2f).</li> <li>3. Objects with low-resolution.</li> </ol>	
<b>Camera, User Interface, and Lighting</b>	Images: 26
(Non-Temporal → Invalid value change)	
<ol style="list-style-type: none"> <li>1. Camera issues such as clipping inside objects or improper character views.</li> <li>2. In-game menus displaying incorrect elements.</li> <li>3. Shadows or lighting effects that do not match the environment.</li> </ol>	

## 4.4 Experiments

### 4.4.1 Experimental setup

**Formulating Questions:** We designed `GLITCHBENCH` as a free-text response benchmark, in contrast with traditional LMM benchmarks that utilize Yes/No or multiple-choice formats [46, 93]. We ask models to describe the unusual aspects of an image by answering three questions:

**(Q1)** *What is unusual about this image?*

**(Q2)** *What is wrong with this image?*

**(Q3)** *Describe the image in detail*

Note that we do not explicitly use the word *glitch* in the question, and we use simple language similar to what a layperson would use. During the inference, we allow models to come up with their own reasoning, and after the model generates the full response, we record it for further evaluation and comparison with the ground truth.

The rationale for free-text answers is that including an ‘unusual’ event description among choices hints to the LMM, letting it answer while disregarding visual aspects.

We included question **Q3** to assess whether the models can accurately report any glitches or unusual elements within the image in extensive captioning. Essentially, this question serves as a visual perception test, evaluating whether the models can identify and describe unusual aspects of the image in a more relaxed condition. For example, in the sample shown in Fig. 4.1, we test the model to see if it can identify the presence of rain in the room. In this case, it indicates that it is raining outside.

**Evaluation:** Following recent successes [212, 226, 107, 20] we employ a language model as a judge to evaluate the model’s responses. We use Llama-2-70B-Chat [186] to compare the model-generated text with the ground truth and determine whether

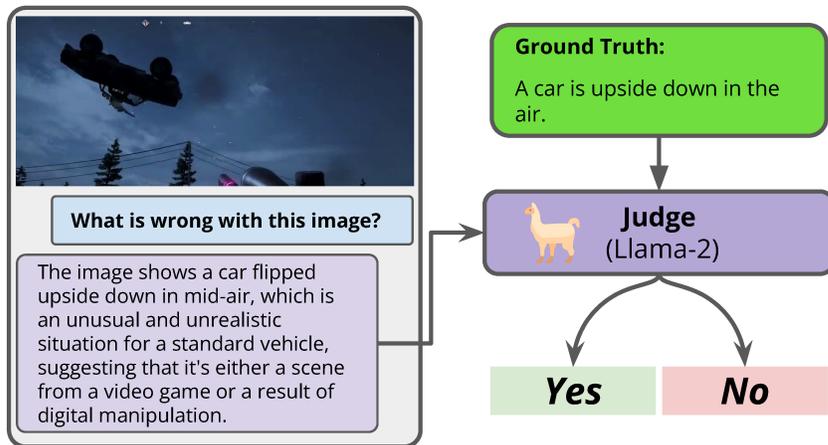


Figure 4.3: To evaluate a model’s response, we ask a judge (the Llama-2-70b-Chat model) to compare it semantically with the ground truth.

the text conveys the same meaning or mentions the event highlighted by the ground truth (see Fig. 4.3).

We report the accuracy of each model on each tested question and present the average performance for **Q1** and **Q2** as the final benchmark result. **Q3** serves as the visual perception test, and we report the performance of the models on it separately.

To assess Llama-2’s judgment and determine if it can effectively serve as an evaluator, we manually reviewed a subset of responses for each model. For each model, we manually labeled 20 samples, with a total of 220 samples.

**Models:** In total, we evaluated 11 LMMs, including GPT-4V [138], and 10 open source models: LLaVA-1.5 (7B and 13B) [108], SPHINX (7B and 13B) [104], InstructBLIP (7B and 13B) [36], Qwen-VL-Chat (10B) [11], MiniGPT-v2 (7B) [26], OtterHD [92], and Fuyo (8B) [14]. We used the default temperature and top-p configurations provided with the model and API. We increased `max_token` to get full responses from models. (See Appendix A.1 for details).

## 4.4.2 Quantitative results

Table 4.2 shows the performance of all the tested models for the three questions. The **Average** performance on **Q1** and **Q2** is the main result of our benchmark. GPT-4V is the best-performing model, achieving 57.2% (**Q1**) and 29.5% (**Q2**) and an average of 43.4%. Next, LLaVA-1.5-13B achieves an average of 35.5% and is the best performing open-source model. These findings show **GlitchBench** is challenging for even state-of-the-art commercial & open-source models.

The performance of GPT-4V on glitch-free images is much higher than on glitch images, with an average accuracy of 91.6%, which suggests that glitch-free images are much easier to handle.

Models exhibit different performance depending on the questions being asked, but all except for the SPHINX family show better performance when prompted with **Q1**. Nevertheless, the gap in performance varies, with GPT-4V showing the largest gap of 27.7pp (57.2% vs. 29.5%). These results highlight that different prompts steer the behavior of LMMs differently and suggest that multi-step reasoning [199, 83] could also help LMMs.

Our results also highlight that higher resolutions improve the performance. In particular, SPHINX-13B, which operates at a higher resolution than SPHINX-7B ( $448 \times 448$  vs.  $224 \times 224$ ), on average performs **+2.9** pp (27.9% vs. 25.0%) better than the base model. Similarly, OtterHD, which employs Fuyu as the base model with enhanced flexibility and support for higher image resolutions, outperforms Fuyu on average by **+15.5** (24.0% vs. 8.5%).

Asking LMMs to extensively caption the image using **Q3** only triggers GPT-4V to produce a very verbose response. In many cases, GPT-4V describes many details in the image and can touch upon the unusual aspects of the image. In this setup, GPT-4V can achieve 64.9%, which is an increase of **+7.7** over **Q1** and **+21.5** pp better than the benchmark results. This gap suggests that GPT-4V can *see* many details in

the image, but it cannot easily focus on the unusual aspects in the frame, indicating a gap in its reasoning capabilities across different modalities and prompts.

**Human evaluation:** Table 4.3 shows the results of comparing between Llama-2 judgments and human evaluations, with the level of agreement for each model measured by Cohen’s Kappa [33]. Cohen’s Kappa demonstrates varying levels of concordance for each model. GPT-4V (0.80), InstructBLIP-7B (0.83), and Qwen-VL (1.00) exhibit substantial to perfect agreement. In contrast, OtterHD (0.50) had fair agreement, and Fuyu (-0.09) shows less than chance agreement, suggesting significant discrepancies. Overall, on all models except for Fuyu, we found above moderate agreement between Llama-2 and human judgment, while on six models, this agreement is substantial.

Table 4.2: Accuracy of various LMMs on `GlitchBench`. Numbers highlighted in   represent the average results of Q1 and Q2, which are the main results of the benchmark. Numbers related to Q3 serve as a visual perception test to measure the ability of models to report glitches in a relaxed manner. Numbers highlighted in   show the maximum agreement achievable with ground truth as perceived by Llama-2’s judgment (%). Numbers highlighted in   represent the results obtained from GPT-4V on glitch-free images.

Question	GPT-4V	LLaVA-1.5		SPHINX		InstructBLIP		OtterHD	Qwen	MiniGPT	Fuyu	
	[138]	[108]		[104]		[36]		[92]	-VL [11]	-v2 [26]	[14]	
	n/a	n/a	7B	13B	7B	13B	7B	13B	8B	10B	7B	8B
Q1. What is unusual about this image?	88.2	57.2	35.2	36.3	19.2	25.3	25.3	21.9	24.8	21.2	19.1	8.6
Q2. What is wrong with this image?	95.5	29.5	23.9	34.7	30.9	30.5	13.8	8.9	23.3	9.3	17.9	8.4
<b>Average</b>	91.6	43.4	29.6	35.5	25.0	27.9	19.6	15.4	24.0	15.2	18.5	8.5
Q3. Describe the image in detail.	-	64.9	28.0	30.5	17.5	21.9	16.0	11.8	21.6	14.0	16.0	7.6
<b>Maximum Agreement</b>	95.5	64.9	35.2	36.3	30.9	30.5	25.3	21.9	24.8	21.2	19.1	8.6

**Accuracy breakdown by category of glitches:** Fig. 4.4 shows the breakdown of the performance of all tested models across the four studied glitch categories. GPT-4V is the best-performing model across all categories, with the exception of the *Rendering and Texture* category, where LLaVA-1.5-13B slightly outperforms it

Table 4.3: Evaluating a subset of responses for comparing Llama-2 with human judgments: Llama-2 and humans exhibit moderate to substantial agreement on all models except for Fuyu.

Model	Llama-2	Human	$\kappa$
GPT-4V	60.0	50.0	0.80
LLaVA-1.5-13B	25.0	20.0	0.57
LLaVA-1.5-7B	35.0	15.0	0.49
Long-SPHINX	25.0	35.0	0.53
SPHINX	30.0	25.0	0.63
InstructBLIP-13B	20.0	10.0	0.62
InstructBLIP-7B	20.0	15.0	0.83
MiniGPT-v2	10.0	5.0	0.64
Qwen-VL	20.0	20.0	1.00
OtterHD	25.0	10.0	0.50
Fuyu	20.0	5.0	-0.09
$\mu \pm \sigma$	$26.4 \pm 12.8$ ,	$19.1 \pm 13.5$	0.64

by +2.3 (41.0% vs. 43.3%). Overall, the *Animation and Pose* category consistently proves to be the most challenging. This category contains images of characters in unusual poses, distorted body joints, or twisted bodies (see an example in Fig. 4.6).

### 4.4.3 Qualitative observations and analysis

**Failing to reason about unusual aspects of the image:** We observed that in several cases, particularly in open-source models, the model reports phrases such as “*the problem with this image is that it is computer-generated*” or “*this is not an actual scene but a scene from a video game*”, along with similar phrases conveying the same meaning. These phrases suggest that, despite the model’s ability to *see* the content of the image, the language component of the model completely fails to reason about the content of the image.

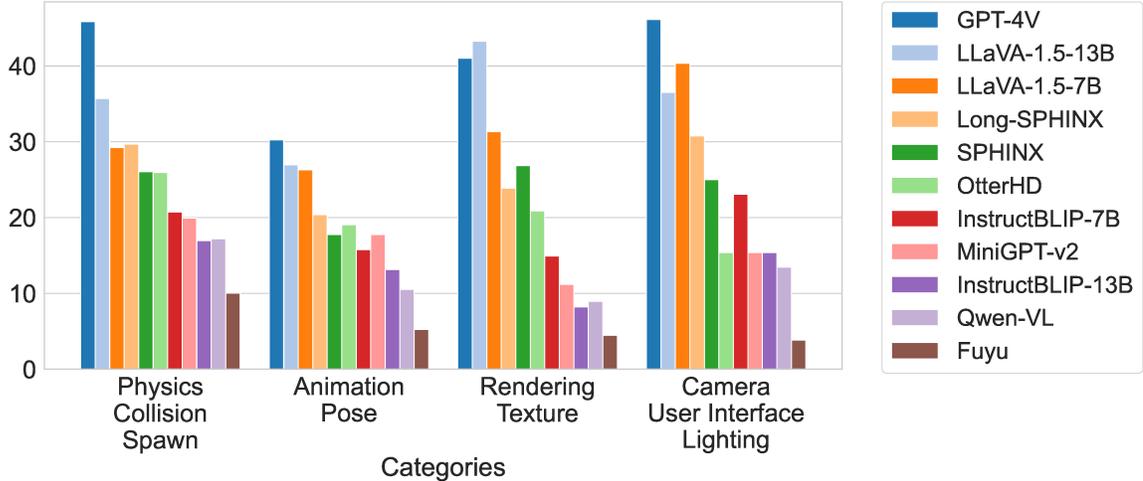


Figure 4.4: The performance (accuracy %) of all tested models on different categories of images in `GlitchBench`.

Another observation is that InstructBLIP-13B often responds with “*nothing*” or similar phrases and completely fails to reason about the image. This is the reason why the smaller InstructBLIP-7B can achieve higher accuracy on `GlitchBench`. (See Appendix A.3.1 for samples.)

**GPT-4V struggles with faces:** GPT-4V is the best-performing model, yet it struggles with characters’ faces, as shown in Fig. 4.5. We found several issues when processing glitches related to faces, and in the majority of cases, GPT-4V fails to detect the glitch and sometimes hallucinates about characters wearing costumes (Fig. A.2), where there are basically no discernible facial features. On the other hand, smaller open-source models can sometimes detect glitches where GPT-4V fails, but they cannot describe the glitch clearly. We hypothesize that this might be due to the privacy features of GPT-4V, preventing it from seeing the face clearly (see Appendix A.3.2 for more samples).

**Poor performance in the *Animation and Pose* category:** Fig. 4.4 shows that *Animation and Pose* is the hardest category of glitches. During manual analysis, we found that LMMs struggle to detect unnatural body and limb configurations

GPT-4V fails to recognize the absence of facial skin



**Question:** What is wrong with this image?

**Ground truth:** The character's face mesh is not rendered correctly.

**GPT-4V:** The image appears to be from a video game or some form of CGI (computer-generated imagery) with very high-quality rendering. The characters depicted are very lifelike, suggesting a significant level of detail and investment in graphics. However, there's nothing inherently "wrong" with the image in terms of content; it seems to portray a close-up moment between two characters.

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth. While it acknowledges the high quality of the graphics and the lifelike nature of the characters, it does not address the issue of the character's face mesh not being rendered correctly, which is the main focus of the ground truth.

Figure 4.5: One of the several cases in which GPT-4V fails to detect a problem with facial features.

and incorrect animations being displayed. For instance, Fig. 4.6 shows an example of an unnatural arm position for a basketball player that GPT-4V cannot detect. This category can be further divided into three subcategories: (1) Heavily distorted body, *e.g.*, when a character’s body is overstretched or expanded; (2) Nuances in body configuration, such as contorted or twisted limbs and hands (Fig. 4.6); and (3) Characters playing animations with invisible props (*e.g.*, a missing gun in the hand, as shown in Fig. 4.2e).

**Prevalent hallucination in open-source models:** *Hallucination* typically refers to situations in which the model’s generated text contains information not present in the image [159, 225, 231, 34]. We noticed that open-source models often hallucinate extra objects or content in the image, *e.g.*, we found that Fuyu’s responses almost always contain hallucinations (see Appendix A.3.4). The hallucination can contain additional characters or entirely new objects. These extra elements sometimes mislead Llama-2 into accepting an incorrect response as correct (see Appendix A.3.5).

**Detecting some glitches requires paying attention to small details:** Different glitches in our dataset require varying levels of visual attention. For example, when a car is flying in the air (Fig. A.11), it usually occupies a large portion of the pixels on the screen, and models can easily pick up on such issues. This type of glitch is easier to catch, and GPT-4V, in particular, is very accurate at detecting it.

Some types of glitches require close attention to detail, such as clipping issues with clothing, where parts of the clothes intersect with the face or body of a character. While GPT-4V is generally the best model for detecting clipping, it is still not perfect. In some cases, GPT-4V misses the clipping, and in other cases, it hallucinates about clipping (see Appendix A.3.6 for samples).

## GPT-4V fails to detect unusual body configuration



**Question:** What is unusual about this image?

**Ground truth:** The left hand of the basketball player is in an unusual position and is stretched backward.

**GPT-4V:** The image appears to be a screenshot from a basketball video game, possibly from the NBA 2K series. The unusual aspects of this image include: One of the players has a bright orange beard that stands out from the rest. Another player seems to be kissing or blowing at the basketball, which is not a typical basketball move or gesture. The characters and graphics look computer-generated, which indicates it's from a video game rather than a real-life game. The interface at the bottom of the image shows various controls and buttons, which further suggests that it's from a video game.

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth. The ground truth focuses on the unusual position of the basketball player's left hand, while the generated description highlights three unusual aspects of the image, none of which relate to the player's hand position.

Figure 4.6: The image shows a basketball player with an unnatural, impossible elbow pose. GPT-4V fails to focus on small details such as body configuration and is unable to report this issue.

## 4.5 Discussion and limitation

**Comparing GlitchBench with other benchmarks:** Tab. 4.4 shows the performance of various models across different benchmarks, which shows that GPT-4V has different performance against open-source models compared to GlitchBench. E.g., LLaVA-1.5 and QWEN-VL score **+5.8** (80.0% vs 74.2%) and **+5.3** pp (79.5% vs 74.2%) higher than GPT-4V on VQAv2. On GlitchBench they lag behind by **-9.9** (33.4% vs. 43.5%) and **-28** pp (15.4% vs. 43.4%). The largest gap is seen in Fuyu’s performance against GPT-4V: while Fuyu exceeds on both OKVQA and AI2D, it significantly lags behind on GlitchBench with only 8.5% compared to GPT-4V’s 43.4%.

In sum, across multiple existing LMM benchmarks, open-source models can perform on par with or even surpass GPT-4V. However, their performance on GlitchBench, which is derived from a real-world task in game quality assurance, falls significantly short of GPT-4V. In other words, the performance of models in real-world settings does not correlate well with existing benchmarks. This discrepancy partly comes from the design choices typical of LMM benchmarks, as they often opt for Yes/No or multiple-choice formats [93, 114, 46]. These formats allow models to find shortcuts for scoring high without necessarily generalizing well to other tasks.

**Limitation:** We constructed our dataset by randomly sampling videos and observed a prevalence of video games with an open-world genre on the Reddit website. Consequently, during our sampling process, video games from this genre, characterized by their distinct mechanics, were more frequently represented compared to other types.

## 4.6 Conclusion

We introduce GlitchBench, a new challenging benchmark for evaluating multimodal models on the video game glitch detection task. Detecting glitches requires various

Table 4.4: Comparing accuracy of models on `GlitchBench` and other visual benchmarks — the bold numbers show the best model per benchmark (%)

Model/Task	Glitch (Ours)	VQAv2 [54]	OKVQA [165]	AI2D [79]	SEED [93]	POPE [99]	MMB [114]
<b>GPT-4V</b>	<b>43.4</b>	74.2	60.6	64.5	-	-	-
<b>LLaVA</b>	33.5	<b>80.0</b>	-	-	70.7	-	<b>67.7</b>
<b>SPHINX</b>	27.9	-	-	-	71.6	<b>90.8</b>	67.1
<b>InstructBLIP</b>	19.6	62.1	-	-	-	78.9	36.0
<b>MiniGPT</b>	18.5	-	57.0	-	-	-	-
<b>QWEN-VL</b>	15.4	79.5	58.6	62.3	58.2	-	60.6
<b>OtterHD</b>	15.2	-	-	-	-	86.1	58.5
<b>Fuyu</b>	8.5	77.4	<b>63.1</b>	<b>73.7</b>	-	-	-

levels of reasoning skills, such as an understanding of the laws of physics and commonsense, making it well-suited for testing the generalization capabilities of large multimodal models. Comparing models’ performance on various multimodal benchmarks and `GlitchBench` reveals a disparity: High performance on prior benchmarks does not guarantee high performance on real-world tasks that demand extensive reasoning abilities. We show that `GlitchBench`, derived from real-world video game quality assurance, presents a new challenge for the AI community and is a valuable addition to existing multimodal benchmarks.

# Chapter 5

## VIDEOGAMEBUNNY: Towards vision assistants for video games

**Bibliographic information:** This chapter was published as: “VIDEOGAMEBUNNY: Towards Vision Assistants for Video Games,” [173] in *the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) 2025*.

Large multimodal models (LMMs) hold substantial promise across various domains, from personal assistance in daily tasks to sophisticated applications like medical diagnostics. However, their capabilities have limitations in the video game domain, such as challenges with scene understanding, hallucinations, and inaccurate descriptions of video game content, especially in open-source models. This chapter describes the development of VIDEOGAMEBUNNY, a LLaVA-style model based on Bunny, specifically tailored for understanding images from video games. We release intermediate checkpoints, training logs, and an extensive dataset comprising 185,259 video game images from 413 titles, along with 389,565 image-instruction pairs that include image captions, question-answer pairs, and a JSON representation of 16 elements of 136,974 images. Our experiments show that our high quality game-related data has the potential to make a relatively small model outperform the much larger state-of-the-art model LLaVa-1.6-34b (which has more than 4x the number of parameters). Our study paves the way for future research in video game understanding on tasks such as playing, commentary, and debugging.

Code and data are available at: <https://videogamebunny.github.io/>

## 5.1 Introduction



Figure 5.1: VIDEOGAMEBUNNY is a model specifically fine-tuned on video game content, enabling it to understand game contexts and respond to related questions more accurately.

The video game industry is projected to be valued at \$321 billion by 2026 [44, 158] and continues to contribute more to the global economy. Generative Artificial Intelligence (GenAI) [74] is rapidly spreading across various sectors, disrupting the ways in which many traditional tasks are executed [21, 41]. In the realm of gaming, GenAI has the potential to enhance many aspects, such as providing a better in-game experience by generating more realistic conversations with non-playable characters (NPCs) [135] or better graphical assets [149, 67, 220, 30].

Large language models (LLMs) and large multimodal models (LMMs) represent advancements in GenAI with the potential to function as vision assistants and solve complex problems across various domains [2, 182, 7]. In video games, LMMs can offer significant benefits for both in-game experiences and game development. In-game, LMMs can serve as vision assistants, enhancing players’ experiences by guiding them through tasks like crafting new items [127]. They also have the potential to narrate the game, summarize events, and highlight critical gameplay moments [6]. For game development, LMMs have the potential to assist in detecting bugs [174], creating bug reports, and deploying automated in-game bots that interact with the environment [180]. These applications require robust multimodal models capable of understanding and interacting with complex game content.

Despite advances and promises, existing LMMs, particularly open-source models, encounter challenges in accurately understanding game content, such as scenes and world physics [174] (e.g., see Fig. 5.1).

In this study, we make the first important step towards addressing these challenges by releasing a suite of datasets specifically designed for video game content and introducing VIDEOGAMEBUNNY, a model trained for video game content understanding. Our study centers on the following research questions:

**(RQ1)** *Which type of data has the potential to improve the model’s performance?*

**(RQ2)** *Which data type mixture strategy improves the model’s performance the most?*

**(RQ3)** *How does VIDEOGAMEBUNNY perform compared to state-of-the-art (SOTA) open-source models on game understanding tasks?*

Our main contributions are as follows:

1. We release VIDEOGAMEBUNNY, a model specifically fine-tuned for video game question-answering tasks.
2. We release a suite of datasets containing 185,259 video game images from 413

games, featuring various gameplay elements and graphical styles. Our datasets include 389,565 image-instruction pairs with captions, question-answering tasks, and JSON representations of images (see Sec. 5.4).

3. We conduct experiments to demonstrate the effectiveness of different instruction datasets and their impact on the model’s performance (see Sec. 5.6).
4. We release a replication package containing the training logs and intermediate checkpoints.

## 5.2 Background and related work

### 5.2.1 Large multimodal models

Large multimodal models (LMMs) enhance large language models (LLMs) by incorporating additional modalities such as images or audio, enabling them to process multimodal inputs and generate textual outputs. The role of the language model is to comprehend user instructions and produce responses based on the additional modality inputs provided. Standard approaches to create LMMs involve combining pre-trained models with different modalities via projection layers. These layers can be implemented using simple mechanisms such as multilayer perceptrons (MLP) [95, 109] or transformer layers [96]. Alternatively, a resampler module like Perceiver [72, 4, 87] or Qformer [37, 232] selectively chooses features to reduce the number of visual tokens based on the context and instruction, enhancing efficiency and maintaining performance.

In this study, we focus on LMMs that accept input images and text to produce responses, particularly using the LLaVA-style architecture [113], which is one of the most popular methods [110, 101, 61, 129]. This architecture employs an MLP layer to integrate vision tokens with a language model.

### 5.2.2 Instruction following data

Large models trained on massive corpora of text, such as GPT-3 [24], T5 [153], and PaLM [32], are not inherently instruction-following, meaning they do not respond to user queries. To enable these models to follow user instructions and answer queries, they usually undergo a process called instruction tuning [139, 222]. This process involves fine-tuning the models to handle specific user instructions, such as questions or commands, allowing them to respond appropriately based on the given instructions.

In the multimodal context, particularly for models that accept visual inputs, there are various types of visual instruction-following data, such as detailed descriptions, conversational style question answering (Q&A), and complex reasoning. Researchers have explored diverse approaches to generate such data, including the use of academic text-oriented visual Q&A datasets [37]. The LLaVA model [95] demonstrated that leveraging a strong text-only LLM and an image dataset annotated with object names and bounding box information can be converted into effective visual instruction-following data.

### 5.2.3 LLMs and LMMs in video games

LLMs have shown strong promise for integration with games for a wide range of tasks, from content creation to game-playing agents [207, 69, 47, 205, 223, 121, 166, 178, 195, 197, 198, 228]. Large multimodal models (LMMs) can further enhance this integration by providing richer context inputs such as images and videos to enable broader applications. Projects like Cradle [180], which focuses on playing Red Dead Redemption 2 with GPT-4V [2] showcase LMMs’ abilities to identify objects, characters, and environmental features and assist in controlling the game. Beyond gameplay, LMMs have found applications in game testing [174, 178], where they are leveraged for detecting and interpreting video game bugs.

Our study is the first to explore enhancing an LMM’s general game understanding, rather than focusing on a specific game or task. We use screenshots from 413

games, aiming to improve capabilities across various game-related tasks by developing broader game comprehension skills.

#### 5.2.4 Empirical analysis of large multimodal models

Some previous studies have conducted experiments to see how different architectural components or data sources affect the general performance of large multimodal models [185, 129, 91, 88]. For example, McKinzie et al.[129] found that the input resolution of the input image plays a crucial role in improving performance, and Laurençon et al.[88] found that utilizing cross-attention between image and language is more effective than the adapter-based method.

We are the first to systematically investigate the impact of different instruction-following datasets and their combinations on the performance of LMMs in game understanding tasks.

### 5.3 VIDEOGAMEBUNNY model architecture

In this section, we describe the architectural choices and configurations behind our model, VIDEOGAMEBUNNY. VIDEOGAMEBUNNY is based on Bunny [61], a family of efficient and high-performing LLMs known for their competitive or superior performance on various benchmarks compared to many open-source alternatives.

Bunny follows the same principle as LLaVA [95, 109] for the integration of image inputs. Using a shallow network of multilayer perceptrons (MLPs) as the projection layer, vision embeddings extracted from a strong pre-trained vision model are processed and provided as image tokens for the language model. This technique effectively leverages pre-trained vision and language models, allowing them to work together efficiently.

Bunny offers various combinations of vision and language models and supports images with resolutions up to  $1152 \times 1152$  pixels. For creating VIDEOGAMEBUNNY, we selected Bunny configurations that deliver the best performance [61] while being

small enough to run on a consumer-grade graphics card. We use LLama-3-8B [130] as the language model and SigLIP [217] with the S<sup>2</sup> wrapper [167] for the vision encoder.

Object framing and zooming inside different parts of the image play a crucial role in image classification [179]. The S<sup>2</sup> wrapper implements this concept by extracting features from an input image at various scales to form a multi-scale feature. This multi-scale feature extraction is particularly useful for video games, which often contain visual elements at different scales, ranging from tiny UI icons to large objects. Furthermore, the S<sup>2</sup> wrapper is particularly more suitable than MEMO [221] since it does not require multiple forward passes during inference and can be automatically integrated into existing LMMs. Fig. 5.2 shows the architecture of VIDEOGAMEBUNNY.

## 5.4 Instruction-following data for video game content

One of the main challenges limiting the ability of open-source models to generalize effectively to video game content is the lack of instruction-following data specific to video games in public datasets. Our goal is to collect game-specific data to address this challenge. In this section, we explain the process of collecting and generating game-specific instruction-following data.

### 5.4.1 Video game images

We collect images from YouTube by searching for *gameplay walkthroughs* with *Full-HD*, *4K*, and *8K* quality. These high-resolution videos ensure that downsampled frames retain more information and details compared to lower quality videos. We randomly sample frames from the downloaded videos and label them with the corresponding game name. In total, our dataset contains 185,259 images from 413 different video games, encompassing various genres, graphic styles, and gameplay mechanics. Fig. 5.3 shows some sample images from our dataset, and Fig. B.5 shows the distribution of images per game.

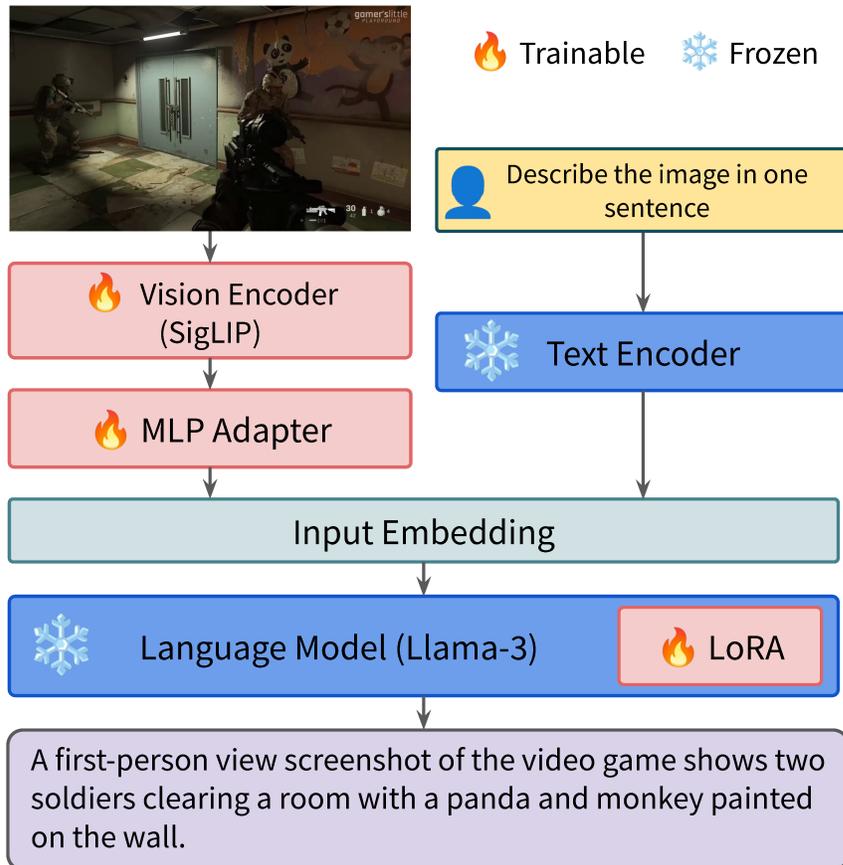


Figure 5.2: Architecture overview of VIDEOGAMEBUNNY. An image input and a textual instruction are fed into the language model to produce a response. The image is passed through a separate pre-trained vision encoder and a projection layer to align the embedding space between the two models. 🔥 and ❄️ icons show trainable and frozen layers respectively

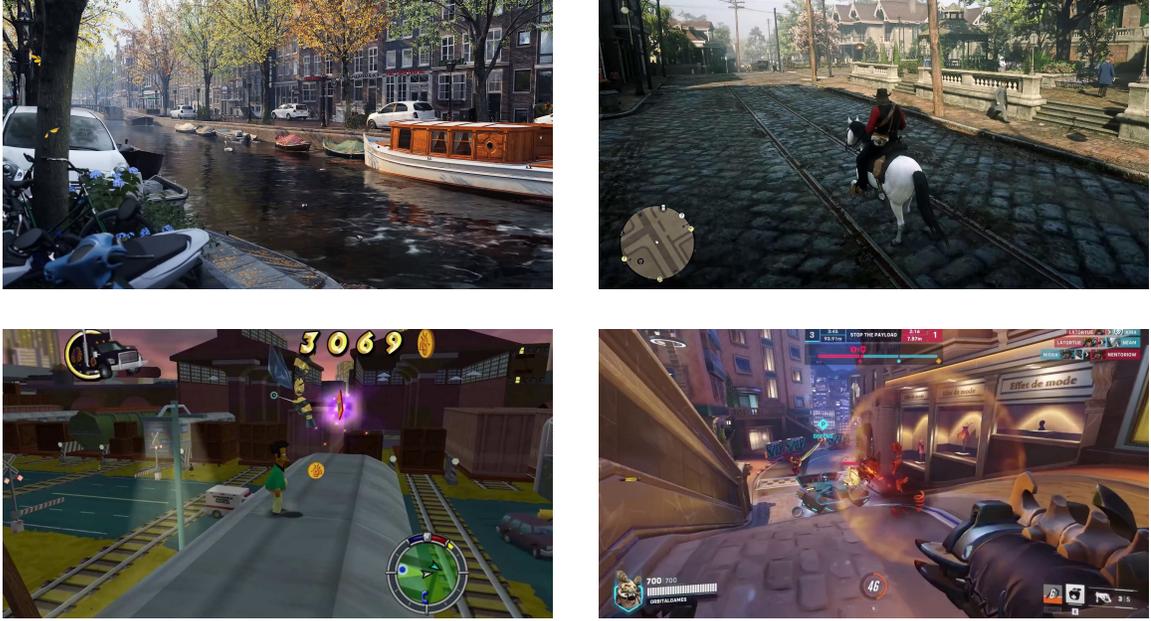


Figure 5.3: Our dataset includes sample video game images that showcase a wide range of characters, environments, mechanics, camera viewpoints, and artistic styles. These styles vary from western to contemporary and futuristic, and from realistic to fantasy settings.

### 5.4.2 Generating instructions

Following previous studies [113, 109, 206], we employ another robust model to generate instructions in the form of user queries and responses for images in our dataset. We categorize the instructions into four types: *short captions* (70,673 samples), *long captions* (70,799 samples), *image-to-JSON* (136,974 samples) and *image-based question answering* (81,122 samples). In this section, we explain how we generate each type of instructions. Fig. 5.4 shows an overview of the data generation process.

#### Image captioning

Image captioning is a basic form of instruction-following that generates a description of the input image. An image caption can be short and concise, providing a high-level overview of the image, or very detailed, covering fine-grain details. Our dataset includes both forms of image captioning to meet user queries, whether they seek a detailed caption or a short summary. In addition, it includes a structured version

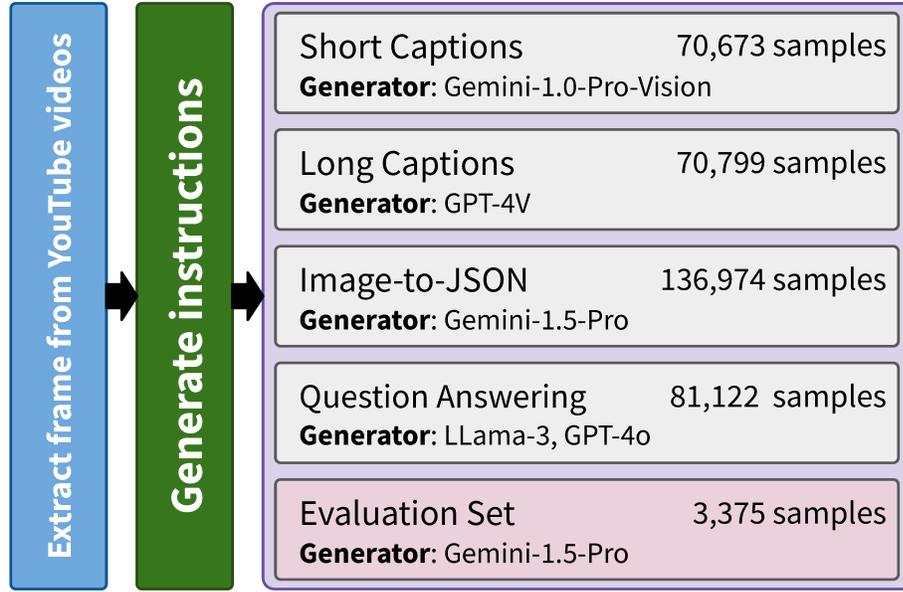


Figure 5.4: Overview of the dataset generation process and models used to generate each type

where the image is described in 16 predefined fields.

**Short captions:** We use Gemini-1.0-pro-vision to obtain short descriptions of a subset of images in our dataset, which includes 70,673 images. We use the “*Describe the image*” prompt, which generates captions typically consisting of one or a few sentences.

**Long captions:** While short captions provide a high-level overview of the image, it lacks many details in the image which might be useful for the user. To address this, we use GPT-4V to get detailed captions of all images in the previous section (see Fig. B.1 for the used prompt).

**Image-to-JSON:** Another comprehensive method for describing images is converting them into a JSON format. This approach summarizes an input image into a JSON structure, with each key describing an element from the image, such as *characters in the image* or *description of Game UI*. Unlike typical captioning, this method provides a template that must be filled. If the image lacks a certain element, that part remains empty, indicating the absence of that element in the image. This ensures a

more detailed and structured representation of the image content.

Another benefit of describing an image in JSON is that this structured representation facilitates integration with other systems. JSON is a widely adopted format for sharing information between different software systems. Summarizing the image as JSON can help integrate LMMs in other systems, such as software testing pipelines, potentially verifying the game output and ensuring that the visual output has the desired properties and information.

To create the image-to-JSON dataset, we use Gemini-1.5-Pro with instructions (Fig. B.2) to convert a given image into a JSON file with hierarchical levels of detail and information. The JSON file contains 16 elements that capture both high-level and fine-grained details of the image. These keys are chosen to capture game-specific elements from the image in isolation, which can be used for downstream applications, such as game testing. It starts with an overall summary of the image and then moves to specific aspects such as detailed character descriptions (including facial expressions and clothing), weather information, summaries of UI and player inventory, objects in the scene, and lighting and environmental effects. Tab. B.1 shows the keys included in the JSON output. Our dataset contains 136,974 Image-JSON pairs. Fig. 5.5 shows a sample of information extracted from an image.

### **Question-Answering conversations**

Moving beyond simple image descriptions, a general and capable model should be able to respond to user questions based on the content of the image. Below, we describe how we generate such data for each image (1) from its long caption and (2) directly from the image itself.

**Llama-3-assisted visual instruction data generation:** We use Llama-3 to convert long captions generated by GPT-4V into a series of question-answering conversations. This approach is similar to the original LLaVA [113] method, but instead of using an object’s name and its bounding box information, we directly utilize long

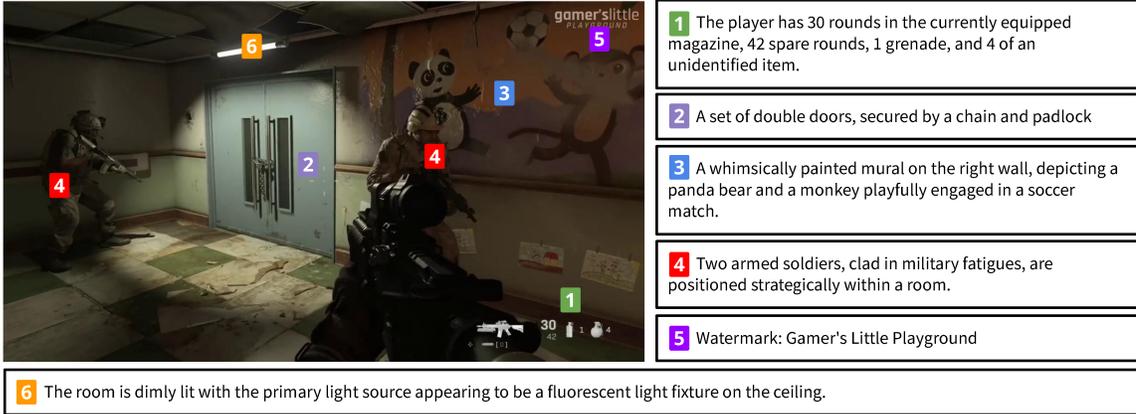


Figure 5.5: Sample information extracted for the image-to-JSON dataset by Gemini-1.5-Pro. Each sample contains detailed information ranging from minor details to high-level descriptions, such as: **1** player inventory, **2** **3** details about the environment, **4** non-player characters, **5** the screenshot’s watermark, and **6** lighting.

captions.

While long captions provide a rich source of information, they lack the structure of question-answer formats. For example, if a caption describes a person in the image with specific details, such as clothing, an LLM can generate a question like, “*What is the color of the dress of the person in the image?*” By utilizing a strong text-only model, we can transform each caption into a multi-turn conversation between a user and an assistant.

We use Llama-3-70B to transform GPT-4V captions into question-answer conversations, with the prompt shown in Fig. B.3. The prompt requires questions to directly relate to the image description. We create 496,469 question-answer pairs for 70,232 images, grouping questions for each image into a multi-turn conversation.

**Image-based question-answering:** We use GPT-4o to generate questions and their answers based on an input image in a *single* prompt. In the prompt (Fig. B.4), we first ask GPT-4o to examine the image and provide a detailed description of its content, then to generate relevant questions based on the content of the image and provide answers for each question. In the prompt, we emphasize that the questions should focus on understanding the image to avoid questions that might not be directly

relevant to the image.

### 5.4.3 Evaluation dataset

To assess model performance on video game understanding tasks, we created a multiple-choice question evaluation set using Gemini-1.5-Pro [182]. This approach allows for an efficient comparison of various models. While Gemini-1.5-Pro offers significant advantages over open-source models for data generation, it does have limitations. We reduce noise in the generated questions as follows:

1. **Initial Generation:** We use Gemini-1.5-Pro to create 4,000 questions across 10 categories related to video game content understanding (see Tab. 5.1).
2. **Quality Assessment:**
  - (a) Self-evaluation: We test Gemini-1.5-Pro on its own questions and found it achieves an accuracy of 84%.
  - (b) Manual validation: A random sampling of questions and answers revealed a 14% error rate (incorrect or indeterminate answers).
3. **Noise Reduction:**
  - (a) We remove 625 samples that Gemini-1.5-Pro had answered incorrectly.
  - (b) We conduct a second manual analysis and found that the error rate dropped to 9%.

We use this evaluation set to measure the performance of the different models we train, allowing us to compare them effectively. During testing, we employ greedy decoding (`temperature = 0`) to ensure that the benchmark results are consistent.

## 5.5 Experiments

In this section, we describe our experiments to explore how our collected instruction-following datasets can improve a model’s understanding of game context. We focus on three research questions:

- (RQ1) *Which type of data has the potential to improve the model’s per-*

Table 5.1: Categories of questions in our dataset, along with a sample question for each category.

Category	Description	Count
<b>Action Understanding</b>	Recognizing and describing the actions taking place within the image. <i>Sample: What action is the character in the foreground performing?</i>	356
<b>Anomalies and Glitches</b>	Identifying errors, bugs, glitches, or placeholder elements within the game environment. <i>Sample: Describe any anomalies or glitches present in the image.</i>	223
<b>Character Analysis</b>	Recognizing characters, understanding their roles, and interpreting their expressions and poses. <i>Sample: What is Aloy's emotional state based on her facial expression?</i>	312
<b>Common Sense Reasoning</b>	Understanding the image using general knowledge and everyday logic. <i>Sample: Based on the score and time remaining, which team is likely to win the match?</i>	430
<b>Gameplay Mechanics</b>	Understanding the rules and mechanics that govern the game. <i>Sample: What game mechanic is most likely being utilized by the player character?</i>	273
<b>OCR and UI</b>	Reading and interpreting on-screen text and user interface elements. <i>Sample: What is written in the caption box at the bottom of the image?</i>	334
<b>Miscellaneous</b>	Any other type of question that does not fit into the previous categories. <i>Sample: What material are the containers in the image primarily made of?</i>	239
<b>Scene Understanding</b>	Recognizing and interpreting the overall environment or setting in the image. <i>Sample: The racetrack depicted in the image is set in what type of environment?</i>	566
<b>Small Details</b>	Identifying and interpreting small but significant details within the image. <i>Sample: What color is the jacket worn by the character in the foreground?</i>	356
<b>Spatial Reasoning</b>	Testing the ability to understand spatial relationships of objects present in the image. <i>Sample: What is the spatial relationship between the two red markers visible in the image?</i>	286

*formance?* In addressing this question, we fine-tune Bunny using a single dataset at a time to observe overall performance trends. Since the primary goal of this experiment is to identify general trends, we fine-tune Bunny on different subset sizes for each dataset only *once*. We increase the subset size from 2K to 60K samples and stop the experiment if we observe a sharp decline in performance.

**(RQ2) Which data type mixture strategy improves the model's performance the most?** We evaluate different data mixing strategies at various sizes to see how both mixture and subset size change the performance of the model. We use the following four strategies:

1. **Random:** We randomly sample without replacement from the combined dataset pool. This serves as a control group, using no specific selection strategy.
2. **Equal:** We select an equal number of samples from each dataset to ensure a balanced representation.

3. **Stratified:** Datasets are mixed based on *video games*, maintaining the game distribution in the final dataset. This balances game representation and ensures diverse image types. We focus on game variety rather than instruction types. Games with insufficient samples are excluded.
4. **Weighted:** We use the three most effective datasets from **RQ1**: image-based question-answering (GPT-4o), long captions, and image-to-JSON. We assign weights: 30% each for GPT-4o and long captions, 40% for image-to-JSON. This prioritizes valuable datasets to assess their impact on model performance.

We fine-tune Bunny on the above dataset mixture strategies with sizes ranging from 2K to 30K. We repeat each experiment three times, using different samples for each strategy to report the mean performance and standard deviation. We stop at 30K since our smallest dataset (generated by GPT-4o) contains 10K samples, and at 30K, we will exhaust the *Equal* and *Weighted* strategies.

**(RQ3) How does VIDEOGAMEBUNNY perform compared to SOTA open-source models on game understanding tasks?** Building on insights from our experiments, we create VIDEOGAMEBUNNY, a model fine-tuned on a dataset of 50K image-instruction samples compiled from all previously introduced datasets. To assess the effectiveness of fine-tuning a smaller model on game-specific data, we evaluate VIDEOGAMEBUNNY against LLaVA-1.6-34b, a SOTA open-source model with 4.2× more parameters.

*Experiment setup:* We instruction tune Bunny with LoRA [68] using the PEFT [125] library. Given that Bunny has been trained on real images, we unfreeze the vision encoder (SigLIP [217]) to adapt to the diverse visual styles of different games. To prevent overfitting and memorization, we fine-tune for only one epoch in all experiments.

Given the importance of reproducibility and accessibility for all researchers, we perform all experiments on a **single NVIDIA A100 GPU (80GB)**, ensuring a balance between computational power and accessibility. The total GPU hours needed

to conduct all experiments, including some preliminary tests, is approximately 900 hours, which is roughly \$2,000 when using cloud providers.

## 5.6 Results

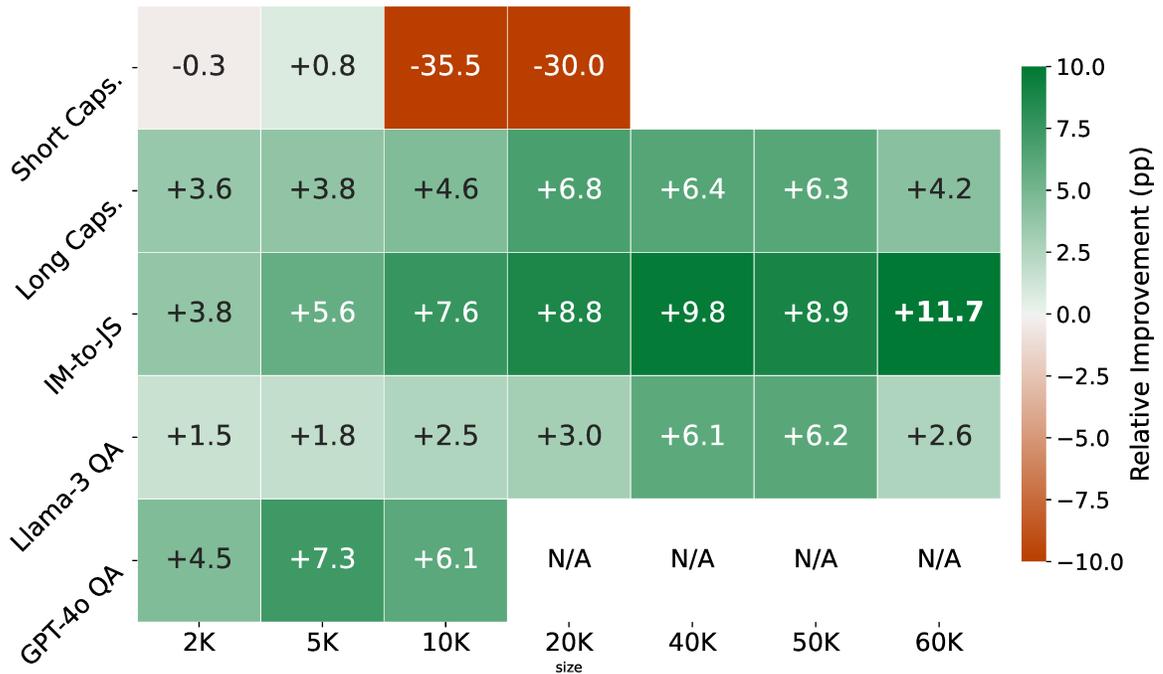


Figure 5.6: Relative accuracy improvement (percentage points) of Bunny fine-tuned on different subsets of each dataset. The image-to-JSON dataset shows a strong positive trend, while the short captions dataset degrades performance. The best performance achieved in the experiment is highlighted in bold.

**RQ1: Which type of data has the potential to improve the model’s performance?**

**The image-to-JSON dataset has the greatest potential to improve the base model’s performance.** Fig. 5.6 shows the performance after fine-tuning Bunny using a single dataset at a time. Fine-tuning on a subset of the image-to-JSON dataset shows the greatest improvements, as this leads to an accuracy above 82% (+8.7 percentage points (pp) above the baseline of 73.3%) for subset sizes over 10K, with the best performance achieved at 60K (+11.7 pp).

While all datasets lead to performance improvement, short captions can degrade it. Fine-tuning Bunny on a dataset of 10K or 20K short captions degrades performance (-35.5 pp and -30 pp), suggesting that short captions do not contain enough signal for the models to improve and can negatively affect the model.

## RQ2: Which data type mixture strategy improves the model’s performance the most?

Table 5.2: Performance (accuracy %) of Bunny on the evaluation set after fine-tuning on a mixture of data with various strategies. The *Weighted* strategy leads to better performance with smaller dataset sizes, but as size increases, all strategies perform similarly. We use a strategy similar to *Weighted* to train VIDEOGAMEBUNNY with 50K samples.

Size	Random	Equal	Stratified	Weighted
2K	76.7 ± 0.9	77.8 ± 0.8	78.0 ± 0.2	79.0 ± 0.6
5K	79.2 ± 0.4	79.9 ± 0.4	80.0 ± 0.5	79.8 ± 0.6
10K	79.8 ± 0.8	80.8 ± 0.6	80.8 ± 0.1	81.4 ± 0.5
20K	81.5 ± 0.1	81.3 ± 0.7	81.8 ± 0.8	82.3 ± 0.9
30K	81.8 ± 0.4	81.2 ± 1.1	81.6 ± 0.7	82.6 ± 0.3
50K	–	–	–	<b>85.1</b>

There is a general improvement trend as we increase the size across all strategies. Tab. 5.2 shows the performance of the models that were fine-tuned using our data mixture strategies. As we increase the dataset size, the mean performance of all mixtures improves. For instance, the *Random* strategy improves from 76.7% at 2K samples to 81.9% at 30K samples and the *Weighted* strategy shows an improvement from 79.0% at 2K samples to 82.6% at 30K samples. This trend demonstrates the value of additional data regardless of the mixing strategy employed.

As the size of dataset increases, different strategies perform similarly. The performance difference between various strategies converges as we increase the

size of the datasets, and they perform similarly in terms of mean and standard deviations. Yet, the *Weighted* method achieves the highest average among other strategies (82.6%). This convergence suggests that the choice of mixing strategy becomes less critical as more data becomes available. In contrast, smaller dataset sizes such as 2k indicate that the *Weighted* strategy outperforms other mixture strategies, achieving an accuracy of  $79.0 \pm 0.6$ .

**Having a uniform distribution of games does not significantly improve performance.** The *Stratified* strategy, which aims to balance the representation of different games in the dataset, does not significantly enhance performance compared to other strategies. For example, in the 2k dataset, the *Stratified* strategy ( $78.0 \pm 0.2$ ) is outperformed by the *Weighted* strategy ( $79.0 \pm 0.6$ ). Similarly, in the 30k dataset, the performance of both strategies is comparable ( $81.6 \pm 0.7$  vs  $82.6 \pm 0.3$ ).

**Fine-tuning improves performance across all categories, with *Anomalies and Glitches* improving the most.** Fig. B.6 shows that fine-tuning improves Bunny’s performance across all categories for almost all dataset sizes. The *Anomalies and Glitches* and *HUD and UI* categories improve the most, with average improvements of  $+32.0$  and  $+21.0$ , using a dataset size of 30K (Tab. 5.3).

### RQ3: How does VIDEOGAMEBUNNY perform compared to SOTA open-source models on game understanding tasks?

VIDEOGAMEBUNNY achieves **85.1%** (Tab. 5.2) on the evaluation set, outperforming all trained models and surpassing various open-source models (Tab. 5.4). It outperforms even LLaVA-1.6-34b, despite its larger parameter count, by  $+1.2$ . Breakdown of accuracy per category reveals that the most significant benefits come from game-specific categories, such as anomaly and glitch detection ( $+16.6$ ) and HUD and UI ( $+3.0$ ) (Tab. B.2). Fig. 5.7 shows a sample from evaluation set where VIDEOGAMEBUNNY correctly answers a common-sense reasoning question while both Bunny and LLaVA-1.6-34b fail.

Table 5.3: Average accuracy improvement (percentage point) for different sizes for each category

Category/ <b>Dataset Size</b>	<b>2K</b>	<b>5K</b>	<b>10K</b>	<b>20K</b>	<b>30K</b>
Action Understanding	1.6	2.5	2.5	3.7	3.9
Anomalies and Glitches	23.4	33.0	33.2	34.0	32.0
Character Analysis	2.6	3.9	4.2	4.7	4.4
Common Sense Reasoning	3.7	4.2	3.8	4.3	4.0
Gameplay Mechanics	4.2	5.0	6.4	8.2	8.9
HUD and UI	9.3	12.9	16.5	18.9	21.0
Miscellaneous	7.2	7.9	9.6	9.9	9.8
Scene Understanding	-0.2	0.6	1.3	2.0	2.0
Small Details	0.3	1.2	2.4	3.4	3.0
Spatial Reasoning	5.3	6.2	7.1	7.8	7.4

We also conduct an informal qualitative analysis, prompting the models with more open-ended questions, which underlines that VIDEOGAMEBUNNY demonstrates an improved understanding of video game content and provides better answers (Fig. 5.8, and Appendix B.3).

Table 5.4: Performance (accuracy %) of various models on the evaluation set.

<b>Model</b>	<b>Accuracy</b>	<b>Model</b>	<b>Accuracy</b>
Bunny-1.1-Llama-3-8B	73.3	LLaVA-v1.5-13b	64.6
VIDEOGAMEBUNNY	<b>85.1</b>	LLaVA-v1.6-vicuna-13b	71.7
LLaVA-v1.5-7b	61.3	LLaVA-v1.6-34b	83.9

## 5.7 Discussion

**Potential negative societal impacts:** Our study initiates the development of a model for understanding game content, with applications in game playing, testing, and commentary. The short-term implications for the gaming industry include en-

## Models comparison in video game understanding



**Question:** Based on the score and time remaining, which team is likely to win the match?

**VIDEOGAMEBUNNY:** (B): The blue team is likely to win ✓

**Bunny:** (C): The red team is likely to win. ✗

**LLaVA-1.6-34b:** (C): The red team is likely to win. ✗

Figure 5.7: VIDEOGAMEBUNNY correctly utilizes information on the HUD to answer the question, while the basemodel Bunny and the larger model LLaVA-1.6-34b fail to provide a correct answer.

## Models comparison in image captioning



**Question:** Write a short summary of the image

**VIDEOGAMEBUNNY:** The image is a screenshot from a video game, showing a character in mid-air, equipped with a wingsuit, flying over a mountainous landscape. The game's HUD elements are visible, including a distance marker and a weapon indicator. . . .

**Bunny:** A video game scene shows a man in a *green shirt* and blue pants jumping off a cliff. He is *holding a parachute* and is in mid-air. . . .

**LLaVA-1.6-34b:** The image appears to be a screenshot from a video game, showing *a first-person perspective* of a character in mid-air, likely after jumping or falling from a height. The character is *equipped with a parachute, which is open* and slowing their descent. . . .

Figure 5.8: VIDEOGAMEBUNNY provides a detailed description of video game content in the image, while other models show inaccuracies (highlighted in *red*). Responses are truncated to save space.

hancing the productivity of game testers and enhancing quality assurance processes. One possible negative impact of such advancement is the facilitation of in-game cheating. As AI models becomes more adept at understanding game contents, there is a risk that this they could be used to create sophisticated cheating tools.

**Performance of VIDEOGAMEBUNNY on GlitchBench:** In Chapter 4, we introduced `GlitchBench`, a dataset containing challenging video game glitches and bugs. This chapter presents a new dataset aimed at fine-tuning models for understanding video game content, along with `VIDEOGAMEBUNNY`, a model built on top of `Bunny`. Evaluating both models on `GlitchBench` shows that the base model, `Bunny`, achieves a performance of 15.2% on `GlitchBench`, while `VIDEOGAMEBUNNY` improves this to 22.8%.

The relatively low performance of these models can be attributed to a significant factor: the nature of the instruction-following datasets originally used to train `Bunny`, hinders the model’s ability to accurately answer queries about video game glitches. Despite `Bunny`’s superior performance compared to other models (e.g., `LLaVA`) on various benchmarks, it struggles to identify video game glitches when prompted with phrases like “*What is wrong with this image?*” Both `Bunny` and `VIDEOGAMEBUNNY` often respond with statements such as “The problem with this image is that it is a screenshot of a video game and not a real photo.”

Addressing this challenge would require either training `Bunny` from scratch with a more diverse mixture of data or including bug-detection specific data for fine-tuning. Both approaches demand substantial resources. Nevertheless, using our dataset, we observe a notable improvement of +5.6pp over the base model.

**Biases and mistakes in dataset generation:** In this study, we use existing models to annotate our dataset. While this approach aligns with previous studies [113, 233] and follows the same principles as the teacher-student learning paradigm [66], we acknowledge that our data may contain biases and mistakes introduced by these existing models.

**Quality of images:** Images in our dataset, sourced from YouTube videos, may include low-quality frames despite filtering for high-quality videos. Motion blur in video games and sampling during scene transitions can result in some blurry or less-than-ideal images.

**Use of various models for annotation:** In the annotation process, we utilize various models depending on their availability and cost. While this process brings diversity to the labels, it is not meant to compare the different models against each other in terms of performance.

**Use of multiple-choice questions to evaluate game understanding:** We use multiple-choice questions to evaluate a model’s game understanding, as it allows for a clear comparison. While this format has been extensively used for benchmarks [63, 214], it might not be the best proxy for game understanding. Future work needs to focus on human evaluation or the use of LLMs as judges [227].

## 5.8 Conclusion

We introduce a new instruction-following dataset, with 389,565 image-instruction pairs, specifically designed for video game understanding. We investigate the effectiveness of fine-tuning LMMs on different instruction-following dataset types and mixtures of them, and finally introduce VIDEOGAMEBUNNY, an 8B parameter model that outperforms a SOTA model, LLaVA-1.6-34b, on a game-related question answering benchmark.

# Chapter 6

## Discussions, conclusion and future work

### 6.1 Discussion

The research presented in this thesis demonstrates the significant potential of foundation models and advanced computer vision techniques in enhancing video game testing and quality assurance processes. The findings across three distinct studies provide valuable insights into the capabilities and limitations of current AI technologies in this domain. While our research shows promise, it is important to acknowledge and address the limitations and potential biases in our approach. The following section discusses the threats to validity that should be considered when interpreting our results and planning future work in this area.

#### 6.1.1 Threats to validity

In this section, we discuss the threats to validity in our research, categorizing them into four main types: internal validity, external validity, construct validity, and conclusion validity.

##### Internal validity

1. **Stochastic nature of models:** The inherent randomness in large language and multimodal models poses a significant threat to the reproducibility of our

results. To mitigate this, we employed greedy decoding with the `temperature` set to zero, aiming to maximize consistency in model outputs. However, even with these precautions, variations may still occur due to different implementation and underlying hardware configurations, and software dependencies (e.g., CUDA drivers, PyTorch versions).

2. **Lack of statistical rigor:** Due to the high computational costs associated with large language models, our studies may lack the statistical power typically expected in empirical software engineering research. This limitation is a common challenge in studies involving these models and may affect the robustness of our findings.
3. **Model variations and instability:** We acknowledge that models can exhibit variations and instability due to a variety of reasons, such as mixture of experts architecture [1] or bugs in various implementations [9, 219]. This instability can affect the consistency and reproducibility of our results.
4. **Limitations of model reasoning:** It is important to note that the large language models used in this study, while appearing to reason, are not actually reasoning in the human sense. Instead, they mimic reasoning based on patterns in their training data. This limitation may affect the interpretation of their outputs and the conclusions drawn from them. As highlighted by Bender et al. [16], these models can be seen as “stochastic parrots,” capable of producing fluent output without true understanding or reasoning capabilities.

## External validity

1. **Generalizability to other games:** Our dataset, particularly in the Game-Physics study, may have a bias towards certain game types (e.g., open-world games). This bias could limit the generalizability of our findings to other game genres or types.

2. **Model availability and versioning:** The potential unavailability of specific model versions in the future (e.g., GPT-4V used in Study 2) poses challenges to the long-term reproducibility of our studies. This impermanence also raises questions about the broader applicability of our findings, as results obtained with current models may not generalize well to future iterations of models.
3. **Specific model versions used:** We used specific versions of models for each part of our study, which may limit the generalizability of our results:
  - For `GlitchBench`: ChatGPT version of GPT-4V in October 2023
  - For `VIDEOGAMEBUNNY`:
    - ChatGPT version of GPT-4V from January 2024 to March 2024
    - GPT-4: `gpt-4-turbo-2024-04-09`
    - Gemini-1.5-Pro: `gemini-1.5-pro-001`
    - Llama-3: We hosted the model using official weights

Results may vary with different versions or implementations of these models.

### **Construct validity**

1. **Limitations of model understanding:** Our interpretation of model outputs and behaviors may not always accurately reflect the underlying processes or capabilities of these models, potentially leading to misinterpretations of their true abilities or limitations.

### **Conclusion validity**

1. **Interpretation of qualitative data:** In analyzing model outputs, especially for open-ended tasks, there's a risk of subjective interpretation influencing our conclusions. We have attempted to mitigate this through multiple reviewers and clear evaluation criteria, but some level of subjectivity remains inherent in such analyses.

2. **Limited sample size:** Due to the resource-intensive nature of working with large language models, our sample sizes may be smaller than ideal for drawing strong statistical conclusions. This limitation is acknowledged and should be considered when interpreting the strength and generalizability of our findings.

By acknowledging these threats to validity, we aim to provide a transparent assessment of the limitations of our research.

## 6.2 Conclusion

This thesis has explored the application of foundation models and advanced computer vision techniques in the realm of video game testing and quality assurance. Our research has focused on leveraging these powerful AI tools to enhance the efficiency and effectiveness of game development processes, particularly in the detection and analysis of visual glitches and anomalies.

This thesis comprises three distinct studies, each contributing to our understanding of how AI can be applied to game quality assurance:

1. **In Chapter 3, we demonstrated that image-text models like CLIP are effective in identifying and matching video game glitches based on textual descriptions.** Our study on searching bug instances in gameplay video repositories revealed the potential of leveraging image-text pre-trained models to overcome the constraints of traditional computer vision approaches in glitch retrieval. By exploring the capability of these models to match textual descriptions of glitches with visual input, we developed a more flexible and adaptable system for identifying a wide range of video game anomalies, without the need for extensive, predefined datasets or rigid classification structures. This finding paves the way for novel approaches in various video game-related tasks and analyses. It offers advancements in video game content analysis in a zero-shot manner, such as for bug clustering and duplicate detection.

2. **In Chapter 4, we found that GPT-4V demonstrates superior performance in glitch detection in video game screenshots and achieves a 64.9% accuracy, while open-source models lag significantly behind.** Our study introducing `GlitchBench`, a new benchmarking dataset for evaluating the efficacy of large multimodal models in detecting video game glitches, revealed a significant performance gap between proprietary and open-source models. While there is still room for improvement, GPT-4V’s performance demonstrates the potential of using large multimodal systems for the task of video game detection. The substantial performance gap between GPT-4V and the best open-source model (LLaVA, achieving only 30.5% accuracy) highlights the need for continued research and development in open-source models for specialized tasks like video game content understanding and glitch detection.
  
3. **In Chapter 5, we introduced VIDEOGAMEBUNNY, a large multimodal model that outperforms LLaVA-1.6-34b on game understanding tasks, with 4.2× fewer parameters.** This chapter explains the development and design choices behind `VIDEOGAMEBUNNY`, a video game-specific large multimodal model. To create `VIDEOGAMEBUNNY`, we introduced a dataset containing 136,974 video game screenshots along with 389,565 image-instruction pairs for training and fine-tuning models for video game content understanding. We investigate the impact of different dataset mixtures on model performance in a series of studies and found that mixture strategies are most important when using small datasets for fine-tuning. As we increase the size, models trained on different strategies converge in performance.

In conclusion, the research presented in this thesis demonstrates the significant potential of foundational model techniques in enhancing video game quality assurance process. These AI-driven approaches can provide promising solutions to the complex challenges faced in game quality assurance, particularly in the detection and analysis

of visual glitches. As the field continues to evolve, further research and development in this area could lead to more efficient, accurate, and automated game testing processes, ultimately contributing to higher quality gaming experiences for users worldwide.

Reflecting on our initial thesis statement, our research has substantiated that AI models can indeed assist in detecting a significant portion of common video game glitches and have the potential to enhance quality assurance processes. While we didn't consistently achieve the 50% detection rate across all studies, our results, particularly with GPT-4V's 64.9% accuracy, demonstrate that this goal is attainable and even surpassable with advanced models. Moreover, VIDEOGAMEBUNNY has proven to be a versatile base model for various video game tasks, including glitch detection. In our internal testing, we found that VIDEOGAMEBUNNY can be easily fine-tuned to detect 45% of common video game glitches in a sample size of 500 publicly reported bugs and synthetic samples that we generated. This performance underscores the potential of specialized AI models to alleviate developer burden and ultimately improve game quality. These findings not only support our thesis but also pave the way for future advancements in AI-assisted game development and quality assurance.

## 6.3 Future work

Future work can be broadly categorized into two different groups: *Model improvements and architectural changes*, which focus on developing novel architectures and new models, and *Applications*, which focus on the more practical integration of current technologies in the video game quality assurance process.

### 6.3.1 Model improvements and architectural changes

This thesis has revealed significant limitations in current large multimodal models, particularly those employing *late-fusion* techniques, which align pre-trained vision and language models through cross-modality projection and instruction-tuning. Such models often exhibit inconsistencies in generalization and cannot understand video

game images that contain glitches. A key issue with these models is their tendency to hallucinate non-existent objects, produce unreliable outputs, and demonstrate that late-fusion has shortcomings for automated video quality assurance applications. This unreliability stems from an over-reliance on the language model component [185], resulting in responses that inadequately incorporate visual information. Such limitations pose significant challenges for accurate and dependable video game glitch detection and automated quality assurance.

To overcome these challenges, future research should prioritize the development of more robust fusion techniques between vision and language modalities. One promising avenue is the exploration of early fusion approaches, exemplified by Meta’s Chameleon model [181]. This method integrates both image and text data concurrently during the pre-training phase, avoiding the isolation of modality-specific training. Such an approach could result in more cohesive and accurate multimodal understanding, potentially mitigating the hallucination issues and improving the model’s ability to provide reliable answers in the context of video game quality assurance.

Additionally, investigating novel attention mechanisms and architectural designs that balance the contributions of visual and textual inputs more effectively could lead to significant advancements in the field.

### 6.3.2 Applications

A promising direction for future work lies in the practical application of foundation models to create tools and visual assistants that enhance the video game testing and quality assurance process. These applications have the potential to significantly improve efficiency, reduce human labor, and contribute to the successful launch of high-quality video games. Key areas for development include:

1. **AI-Powered visual assistants:** Develop tools that leverage foundation models to aid human testers in detecting and reporting video game bugs more efficiently. These assistants could:

- Accelerate bug detection by highlighting potential issues in real-time.
- Automate the generation of detailed bug reports based on tester observations and confirmations.
- Provide intelligent suggestions for additional areas to test based on detected anomalies.

2. **In-Game AI Agents:** Expand on the concept of AI-driven game interaction, similar to projects like Cradle [180], but with a specific focus on game testing:

- Develop agents capable of mimicking human player behavior to systematically explore game environments and mechanics.
- Implement sophisticated bug detection algorithms within these agents to identify visual glitches, gameplay inconsistencies, and performance issues.
- Create agents that can reproduce and document complex bug scenarios, reducing the burden on human testers.

By focusing on these application areas, future research can bridge the gap between theoretical advancements in AI and practical improvements in video game quality assurance. This approach has the potential to revolutionize the game development process, leading to higher-quality products, reduced development costs, and ultimately, more satisfying gaming experiences for end-users.

# Bibliography

- [1] 152334h. *Non-determinism in GPT-4 is caused by Sparse MoE*. Accessed on September 24, 2024. 2023. URL: <https://152334h.github.io/blog/non-determinism-in-gpt-4/>.
- [2] Josh Achiam, Steven Adler, Sandhini Agarwal, et al. “Gpt-4 technical report”. In: *arXiv preprint arXiv:2303.08774* (2023).
- [3] Harsh Agrawal, Karan Desai, Yufei Wang, et al. “nocaps: novel object captioning at scale”. In: *CoRR* abs/1812.08658 (2018). arXiv: 1812.08658. URL: <http://arxiv.org/abs/1812.08658>.
- [4] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, et al. “Flamingo: a Visual Language Model for Few-Shot Learning”. In: *CoRR* abs/2204.14198 (2022). DOI: 10.48550/ARXIV.2204.14198. arXiv: 2204.14198. URL: <https://doi.org/10.48550/arXiv.2204.14198>.
- [5] Michael A. Alcorn, Qi Li, Zhitao Gong, et al. “Strike (with) a Pose: Neural Networks Are Easily Fooled by Strange Poses of Familiar Objects”. In: *CoRR* abs/1811.11553 (2018). arXiv: 1811.11553. URL: <http://arxiv.org/abs/1811.11553>.
- [6] Anonymous. *While I wait for GPT-4o with updated voice capabilities, I decided to create a prototype using multiple open source models to simulate an AI commentator who can see your screen and listen to in-game dialogue*. Reddit r/OpenAI. Accessed: June 30, 2024. 2024. URL: [https://www.reddit.com/r/OpenAI/comments/1dm6lg9/while\\_i\\_wait\\_for\\_gpt4o\\_with\\_updated\\_voice/](https://www.reddit.com/r/OpenAI/comments/1dm6lg9/while_i_wait_for_gpt4o_with_updated_voice/).
- [7] Anthropic. *Introducing the next generation of Claude*. Anthropic, 2024. URL: <https://www.anthropic.com/news/claude-3-family> (visited on 06/20/2024).
- [8] *Assassin’s Creed Unity’s Disappearing Face Glitch*. Reddit. Assassin’s Creed Unity had a glitch causing characters’ faces to disappear, leaving only floating eyes and teeth. 2020. URL: [https://www.reddit.com/r/assassinscreed/comments/ex45yk/ac\\_unity\\_face\\_bugs/](https://www.reddit.com/r/assassinscreed/comments/ex45yk/ac_unity_face_bugs/).
- [9] Berk Atil, Alexa Chittams, Liseng Fu, Ferhan Ture, Lixinyu Xu, and Breck Baldwin. “LLM Stability: A detailed analysis with some surprises”. In: *arXiv preprint arXiv:2408.04667* (2024).

- [10] Anas Awadalla, Irena Gao, Josh Gardner, et al. “OpenFlamingo: An Open-Source Framework for Training Large Autoregressive Vision-Language Models”. In: *CoRR* abs/2308.01390 (2023). DOI: 10.48550/ARXIV.2308.01390. arXiv: 2308.01390. URL: <https://doi.org/10.48550/arXiv.2308.01390>.
- [11] Jinze Bai, Shuai Bai, Shusheng Yang, et al. “Qwen-VL: A Frontier Large Vision-Language Model with Versatile Abilities”. In: *CoRR* abs/2308.12966 (2023). DOI: 10.48550/ARXIV.2308.12966. arXiv: 2308.12966. URL: <https://doi.org/10.48550/arXiv.2308.12966>.
- [12] Max Bain, Arsha Nagrani, Gül Varol, and Andrew Zisserman. “Frozen in Time: A Joint Video and Image Encoder for End-to-End Retrieval”. In: *CoRR* abs/2104.00650 (2021). arXiv: 2104.00650. URL: <https://arxiv.org/abs/2104.00650>.
- [13] Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. “The Pushshift Reddit Dataset”. In: *CoRR* abs/2001.08435 (2020). arXiv: 2001.08435. URL: <https://arxiv.org/abs/2001.08435>.
- [14] Rohan Bavishi, Erich Elsen, Curtis Hawthorne, et al. *Introducing our Multimodal Models*. 2023. URL: <https://www.adept.ai/blog/fuyu-8b>.
- [15] Suzanna Becker and Geoffrey E Hinton. “Self-organizing neural network that discovers surfaces in random-dot stereograms”. In: *Nature* 355.6356 (1992), pp. 161–163. DOI: 10.1038/355161a0.
- [16] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. “On the dangers of stochastic parrots: Can language models be too big?” In: *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*. 2021, pp. 610–623.
- [17] Joakim Bergdahl, Camilo Gordillo, Konrad Tollmar, and Linus Gisslén. “Augmenting Automated Game Testing with Deep Reinforcement Learning”. In: *CoRR* abs/2103.15819 (2021). arXiv: 2103.15819. URL: <https://arxiv.org/abs/2103.15819>.
- [18] Christopher Berner, Greg Brockman, Brooke Chan, et al. “Dota 2 with Large Scale Deep Reinforcement Learning”. In: *CoRR* abs/1912.06680 (2019). arXiv: 1912.06680. URL: <http://arxiv.org/abs/1912.06680>.
- [19] Nicolas Bettenburg, Sascha Just, Adrian Schröter, Cathrin Weiss, Rahul Premraj, and Thomas Zimmermann. “What makes a good bug report?” In: *Proceedings of the 16th ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2008, Atlanta, Georgia, USA, November 9-14, 2008*. Ed. by Mary Jean Harrold and Gail C. Murphy. ACM, 2008, pp. 308–318. DOI: 10.1145/1453101.1453146. URL: <https://doi.org/10.1145/1453101.1453146>.
- [20] Yonatan Bitton, Hritik Bansal, Jack Hessel, et al. “VisIT-Bench: A Benchmark for Vision-Language Instruction Following Inspired by Real-World Use”. In: *CoRR* abs/2308.06595 (2023). DOI: 10.48550/ARXIV.2308.06595. arXiv: 2308.06595. URL: <https://doi.org/10.48550/arXiv.2308.06595>.

- [21] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, et al. “On the Opportunities and Risks of Foundation Models”. In: *CoRR* abs/2108.07258 (2021). arXiv: 2108.07258. URL: <https://arxiv.org/abs/2108.07258>.
- [22] Antonio Borrelli, Vittoria Nardone, Giuseppe A. Di Lucca, Gerardo Canfora, and Massimiliano Di Penta. “Detecting Video Game-Specific Bad Smells in Unity Projects”. In: *MSR '20: 17th International Conference on Mining Software Repositories, Seoul, Republic of Korea, 29-30 June, 2020*. Ed. by Sunghun Kim, Georgios Gousios, Sarah Nadi, and Joseph Hejderup. ACM, 2020, pp. 198–208. DOI: 10.1145/3379597.3387454. URL: <https://doi.org/10.1145/3379597.3387454>.
- [23] Jane Bromley, James W. Bentz, Léon Bottou, et al. “Signature Verification Using A "Siamese" Time Delay Neural Network”. In: *Int. J. Pattern Recognit. Artif. Intell.* 7.4 (1993), pp. 669–688. DOI: 10.1142/S0218001493000339. URL: <https://doi.org/10.1142/S0218001493000339>.
- [24] Tom B. Brown, Benjamin Mann, Nick Ryder, et al. “Language Models are Few-Shot Learners”. In: *CoRR* abs/2005.14165 (2020). arXiv: 2005.14165. URL: <https://arxiv.org/abs/2005.14165>.
- [25] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, et al. “Sparks of Artificial General Intelligence: Early experiments with GPT-4”. In: *CoRR* abs/2303.12712 (2023). DOI: 10.48550/ARXIV.2303.12712. arXiv: 2303.12712. URL: <https://doi.org/10.48550/arXiv.2303.12712>.
- [26] Jun Chen, Deyao Zhu, Xiaoqian Shen, et al. “MiniGPT-v2: large language model as a unified interface for vision-language multi-task learning”. In: *CoRR* abs/2310.09478 (2023). DOI: 10.48550/ARXIV.2310.09478. arXiv: 2310.09478. URL: <https://doi.org/10.48550/arXiv.2310.09478>.
- [27] Ke Chen, Yufei Li, Yingfeng Chen, Changjie Fan, Zhipeng Hu, and Wei Yang. “GLIB: Towards Automated Test Oracle for Graphically-Rich Applications”. In: *CoRR* abs/2106.10507 (2021). arXiv: 2106.10507. URL: <https://arxiv.org/abs/2106.10507>.
- [28] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. “A Simple Framework for Contrastive Learning of Visual Representations”. In: *CoRR* abs/2002.05709 (2020). arXiv: 2002.05709. URL: <https://arxiv.org/abs/2002.05709>.
- [29] Xinlei Chen, Hao Fang, Tsung-Yi Lin, et al. “Microsoft COCO Captions: Data Collection and Evaluation Server”. In: *CoRR* abs/1504.00325 (2015). arXiv: 1504.00325. URL: <http://arxiv.org/abs/1504.00325>.
- [30] Yiwen Chen, Tong He, Di Huang, et al. “MeshAnything: Artist-Created Mesh Generation with Autoregressive Transformers”. In: *CoRR* abs/2406.10163 (2024). DOI: 10.48550/ARXIV.2406.10163. arXiv: 2406.10163. URL: <https://doi.org/10.48550/arXiv.2406.10163>.

- [31] Myung Jin Choi, Antonio Torralba, and Alan S. Willsky. “Context models and out-of-context objects”. In: *Pattern Recognit. Lett.* 33.7 (2012), pp. 853–862. DOI: 10.1016/J.PATREC.2011.12.004. URL: <https://doi.org/10.1016/j.patrec.2011.12.004>.
- [32] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, et al. “PaLM: Scaling Language Modeling with Pathways”. In: *CoRR* abs/2204.02311 (2022). DOI: 10.48550/ARXIV.2204.02311. arXiv: 2204.02311. URL: <https://doi.org/10.48550/arXiv.2204.02311>.
- [33] Jacob Cohen. “A Coefficient of Agreement for Nominal Scales”. In: *Educational and Psychological Measurement* 20.1 (1960), p. 37.
- [34] Chenhang Cui, Yiyang Zhou, Xinyu Yang, et al. “Holistic Analysis of Hallucination in GPT-4V(ision): Bias and Interference Challenges”. In: *CoRR* abs/2311.03287 (2023). DOI: 10.48550/ARXIV.2311.03287. arXiv: 2311.03287. URL: <https://doi.org/10.48550/arXiv.2311.03287>.
- [35] *Cyberpunk 2077’s Launch Issues*. Windows Central. Cyberpunk 2077 experienced numerous bugs and performance issues at launch, particularly on last-gen consoles. 2020. URL: <https://www.windowscentral.com/cyberpunk-2077-known-bugs-and-launch-issues>.
- [36] Wenliang Dai, Junnan Li, Dongxu Li, et al. “InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning”. In: *CoRR* abs/2305.06500 (2023). DOI: 10.48550/ARXIV.2305.06500. arXiv: 2305.06500. URL: <https://doi.org/10.48550/arXiv.2305.06500>.
- [37] Wenliang Dai, Junnan Li, Dongxu Li, et al. “InstructBLIP: Towards General-purpose Vision-Language Models with Instruction Tuning”. In: *CoRR* abs/2305.06500 (2023). DOI: 10.48550/ARXIV.2305.06500. arXiv: 2305.06500. URL: <https://doi.org/10.48550/arXiv.2305.06500>.
- [38] Parmida Davarmanesh, Kuanhao Jiang, Tingting Ou, et al. “Automating Artifact Detection in Video Games”. In: *CoRR* abs/2011.15103 (2020). arXiv: 2011.15103. URL: <https://arxiv.org/abs/2011.15103>.
- [39] Qingxiu Dong, Lei Li, Damai Dai, et al. “A Survey for In-context Learning”. In: *CoRR* abs/2301.00234 (2023). DOI: 10.48550/ARXIV.2301.00234. arXiv: 2301.00234. URL: <https://doi.org/10.48550/arXiv.2301.00234>.
- [40] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *CoRR* abs/2010.11929 (2020). arXiv: 2010.11929. URL: <https://arxiv.org/abs/2010.11929>.
- [41] Tyna Eloundou, Sam Manning, Pamela Mishkin, and Daniel Rock. “GPTs are GPTs: An Early Look at the Labor Market Impact Potential of Large Language Models”. In: *CoRR* abs/2303.10130 (2023). DOI: 10.48550/ARXIV.2303.10130. arXiv: 2303.10130. URL: <https://doi.org/10.48550/arXiv.2303.10130>.

- [42] Patrick Esser, Sumith Kulal, Andreas Blattmann, et al. “Scaling Rectified Flow Transformers for High-Resolution Image Synthesis”. In: *CoRR* abs/2403.03206 (2024). DOI: 10.48550/ARXIV.2403.03206. arXiv: 2403.03206. URL: <https://doi.org/10.48550/arXiv.2403.03206>.
- [43] MohammadAmin Fazli, Ali Owfi, and Mohammad Reza Taesiri. “Under the Skin of Foundation NFT Auctions”. In: *CoRR* abs/2109.12321 (2021). arXiv: 2109.12321. URL: <https://arxiv.org/abs/2109.12321>.
- [44] World Economic Forum. “Gaming boomed in lockdown and market value will reach \$320bn”. In: *World Economic Forum* (2023). URL: <https://www.weforum.org/agenda/2023/01/gaming-market-value-320bn-2026/>.
- [45] Stella Frank, Emanuele Bugliarello, and Desmond Elliott. “Vision-and-Language or Vision-for-Language? On Cross-Modal Influence in Multimodal Transformers”. In: *CoRR* abs/2109.04448 (2021). arXiv: 2109.04448. URL: <https://arxiv.org/abs/2109.04448>.
- [46] Chaoyou Fu, Peixian Chen, Yunhang Shen, et al. “MME: A Comprehensive Evaluation Benchmark for Multimodal Large Language Models”. In: *CoRR* abs/2306.13394 (2023). DOI: 10.48550/ARXIV.2306.13394. arXiv: 2306.13394. URL: <https://doi.org/10.48550/arXiv.2306.13394>.
- [47] Roberto Gallotta, Graham Todd, Marvin Zammit, et al. “Large Language Models and Games: A Survey and Roadmap”. In: *CoRR* abs/2402.18659 (2024). DOI: 10.48550/ARXIV.2402.18659. arXiv: 2402.18659. URL: <https://doi.org/10.48550/arXiv.2402.18659>.
- [48] Pablo García-Sánchez, Alberto Paolo Tonda, Antonio Mora García, Giovanni Squillero, and Juan Julián Merelo Guervós. “Automated playtesting in collectible card games using evolutionary algorithms: A case study in hearthstone”. In: *Knowl. Based Syst.* 153 (2018), pp. 133–146. DOI: 10.1016/J.KNOSYS.2018.04.030. URL: <https://doi.org/10.1016/j.knosys.2018.04.030>.
- [49] Yuying Ge, Yixiao Ge, Xihui Liu, et al. “Bridging Video-text Retrieval with Multiple Choice Questions”. In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*. IEEE, 2022, pp. 16146–16155. DOI: 10.1109/CVPR52688.2022.01569. URL: <https://doi.org/10.1109/CVPR52688.2022.01569>.
- [50] Tejas Gokhale, Hamid Palangi, Besmira Nushi, et al. “Benchmarking Spatial Relationships in Text-to-Image Generation”. In: *CoRR* abs/2212.10015 (2022). DOI: 10.48550/ARXIV.2212.10015. arXiv: 2212.10015. URL: <https://doi.org/10.48550/arXiv.2212.10015>.
- [51] Wendy Gonzalez. *Council Post: How AI Is Cropping Up in the Agriculture Industry*. Feb. 2023. URL: <https://www.forbes.com/sites/forbesbusinesscouncil/2023/02/02/how-ai-is-cropping-up-in-the-agriculture-industry/>.
- [52] Google. *Bard*. 2023. URL: <https://bard.google.com>.

- [53] Camilo Gordillo, Joakim Bergdahl, Konrad Tollmar, and Linus Gisslén. “Improving Playtesting Coverage via Curiosity Driven Reinforcement Learning Agents”. In: *CoRR* abs/2103.13798 (2021). arXiv: 2103.13798. URL: <https://arxiv.org/abs/2103.13798>.
- [54] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. “Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering”. In: *CoRR* abs/1612.00837 (2016). arXiv: 1612.00837. URL: <http://arxiv.org/abs/1612.00837>.
- [55] Jean-Bastien Grill, Florian Strub, Florent Altché, et al. “Bootstrap Your Own Latent: A New Approach to Self-Supervised Learning”. In: *CoRR* abs/2006.07733 (2020). arXiv: 2006.07733. URL: <https://arxiv.org/abs/2006.07733>.
- [56] Nitzan Bitton Guetta, Yonatan Bitton, Jack Hessel, et al. “Breaking Common Sense: WHOOPS! A Vision-and-Language Benchmark of Synthetic and Compositional Images”. In: *CoRR* abs/2303.07274 (2023). DOI: 10.48550/ARXIV.2303.07274. arXiv: 2303.07274. URL: <https://doi.org/10.48550/arXiv.2303.07274>.
- [57] Danna Gurari, Qing Li, Abigale J. Stangl, et al. “VizWiz Grand Challenge: Answering Visual Questions from Blind People”. In: *CoRR* abs/1802.08218 (2018). arXiv: 1802.08218. URL: <http://arxiv.org/abs/1802.08218>.
- [58] Matthew Guzdial, Nicholas Liao, and Mark O. Riedl. “Co-Creative Level Design via Machine Learning”. In: *CoRR* abs/1809.09420 (2018). arXiv: 1809.09420. URL: <http://arxiv.org/abs/1809.09420>.
- [59] Matthew Guzdial and Mark O. Riedl. “Game Level Generation from Gameplay Videos”. In: *Proceedings of the Twelfth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, AIIDE 2016, October 8-12, 2016, Burlingame, California, USA*. Ed. by Nathan R. Sturtevant and Brian Magerko. AAAI Press, 2016, pp. 44–50. URL: <http://aaai.org/ocs/index.php/AIIDE/AIIDE16/paper/view/14008>.
- [60] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. “Deep Residual Learning for Image Recognition”. In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: <http://arxiv.org/abs/1512.03385>.
- [61] Muyang He, Yexin Liu, Boya Wu, et al. “Efficient Multimodal Learning from Data-centric Perspective”. In: *CoRR* abs/2402.11530 (2024). DOI: 10.48550/ARXIV.2402.11530. arXiv: 2402.11530. URL: <https://doi.org/10.48550/arXiv.2402.11530>.
- [62] Dan Hendrycks, Steven Basart, Norman Mu, et al. “The Many Faces of Robustness: A Critical Analysis of Out-of-Distribution Generalization”. In: *CoRR* abs/2006.16241 (2020). arXiv: 2006.16241. URL: <https://arxiv.org/abs/2006.16241>.
- [63] Dan Hendrycks, Collin Burns, Steven Basart, et al. “Measuring Massive Multi-task Language Understanding”. In: *CoRR* abs/2009.03300 (2020). arXiv: 2009.03300. URL: <https://arxiv.org/abs/2009.03300>.

- [64] Dan Hendrycks and Thomas G. Dietterich. “Benchmarking Neural Network Robustness to Common Corruptions and Perturbations”. In: *CoRR* abs/1807.01697 (2018). arXiv: 1807.01697. URL: <http://arxiv.org/abs/1807.01697>.
- [65] Tuomo Hiippala, Malihe Alikhani, Jonas Haverinen, et al. “AI2D-RST: A multimodal corpus of 1000 primary school science diagrams”. In: *CoRR* abs/1912.03879 (2019). arXiv: 1912.03879. URL: <http://arxiv.org/abs/1912.03879>.
- [66] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. “Distilling the Knowledge in a Neural Network”. In: *CoRR* abs/1503.02531 (2015). arXiv: 1503.02531. URL: <http://arxiv.org/abs/1503.02531>.
- [67] Benjie Holson. *Dimension hopper part 1*. June 2023. URL: <https://generalrobots.substack.com/p/dimension-hopper-part-1>.
- [68] Edward J. Hu, Yelong Shen, Phillip Wallis, et al. “LoRA: Low-Rank Adaptation of Large Language Models”. In: *CoRR* abs/2106.09685 (2021). arXiv: 2106.09685. URL: <https://arxiv.org/abs/2106.09685>.
- [69] Sihao Hu, Tiansheng Huang, Fatih Ilhan, et al. “A Survey on Large Language Model-Based Game Agents”. In: *CoRR* abs/2404.02039 (2024). DOI: 10.48550/ARXIV.2404.02039. arXiv: 2404.02039. URL: <https://doi.org/10.48550/arXiv.2404.02039>.
- [70] Drew A Hudson and Christopher D Manning. “Gqa: A new dataset for real-world visual reasoning and compositional question answering”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 6700–6709.
- [71] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, et al. *OpenCLIP*. Version 0.1. If you use this software, please cite it as below. July 2021. DOI: 10.5281/zenodo.5143773.
- [72] Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. “Perceiver: General Perception with Iterative Attention”. In: *CoRR* abs/2103.03206 (2021). arXiv: 2103.03206. URL: <https://arxiv.org/abs/2103.03206>.
- [73] Huaizu Jiang, Xiaojian Ma, Weili Nie, Zhiding Yu, Yuke Zhu, and Anima Anandkumar. “Bongard-HOI: Benchmarking Few-Shot Visual Reasoning for Human-Object Interactions”. In: *CoRR* abs/2205.13803 (2022). DOI: 10.48550/ARXIV.2205.13803. arXiv: 2205.13803. URL: <https://doi.org/10.48550/arXiv.2205.13803>.
- [74] A Jo. “The promise and peril of generative AI”. In: *Nature* 614.1 (2023), pp. 214–216.
- [75] Jeff Johnson, Matthijs Douze, and Hervé Jégou. “Billion-scale similarity search with GPUs”. In: *CoRR* abs/1702.08734 (2017). arXiv: 1702.08734. URL: <http://arxiv.org/abs/1702.08734>.

- [76] Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, et al. “Unity: A General Platform for Intelligent Agents”. In: *CoRR* abs/1809.02627 (2018). arXiv: 1809.02627. URL: <http://arxiv.org/abs/1809.02627>.
- [77] Niels Justesen, Philip Bontrager, Julian Togelius, and Sebastian Risi. “Deep Learning for Video Game Playing”. In: *CoRR* abs/1708.07902 (2017). arXiv: 1708.07902. URL: <http://arxiv.org/abs/1708.07902>.
- [78] Amita Kamath, Jack Hessel, and Kai-Wei Chang. “What’s “up” with vision-language models? Investigating their struggle with spatial reasoning”. In: *CoRR* abs/2310.19785 (2023). DOI: 10.48550/ARXIV.2310.19785. arXiv: 2310.19785. URL: <https://doi.org/10.48550/arXiv.2310.19785>.
- [79] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Min Joon Seo, Hannaneh Hajishirzi, and Ali Farhadi. “A Diagram Is Worth A Dozen Images”. In: *CoRR* abs/1603.07396 (2016). arXiv: 1603.07396. URL: <http://arxiv.org/abs/1603.07396>.
- [80] Faraz Khadivpour and Matthew Guzdial. “Explainability via Responsibility”. In: *CoRR* abs/2010.01676 (2020). arXiv: 2010.01676. URL: <https://arxiv.org/abs/2010.01676>.
- [81] Nazanin Yousefzadeh Khameneh and Matthew Guzdial. “Entity Embedding as Game Representation”. In: *CoRR* abs/2010.01685 (2020). arXiv: 2010.01685. URL: <https://arxiv.org/abs/2010.01685>.
- [82] Prannay Khosla, Piotr Teterwak, Chen Wang, et al. “Supervised Contrastive Learning”. In: *CoRR* abs/2004.11362 (2020). arXiv: 2004.11362. URL: <https://arxiv.org/abs/2004.11362>.
- [83] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. “Large Language Models are Zero-Shot Reasoners”. In: *CoRR* abs/2205.11916 (2022). DOI: 10.48550/ARXIV.2205.11916. arXiv: 2205.11916. URL: <https://doi.org/10.48550/arXiv.2205.11916>.
- [84] Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. “Fine-Tuning can Distort Pretrained Features and Underperform Out-of-Distribution”. In: *CoRR* abs/2202.10054 (2022). arXiv: 2202.10054. URL: <https://arxiv.org/abs/2202.10054>.
- [85] Christoph H. Lampert, Hannes Nickisch, and Stefan Harmeling. “Learning to detect unseen object classes by between-class attribute transfer”. In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 2009, pp. 951–958. DOI: 10.1109/CVPR.2009.5206594. URL: <https://doi.org/10.1109/CVPR.2009.5206594>.
- [86] Hugo Larochelle, Dumitru Erhan, and Yoshua Bengio. “Zero-data Learning of New Tasks”. In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*. Ed. by Dieter Fox and Carla P. Gomes. AAAI Press, 2008, pp. 646–651. URL: <http://www.aaai.org/Library/AAAI/2008/aaai08-103.php>.

- [87] Hugo Laurençon, Lucile Saulnier, Léo Tronchon, et al. “OBELICS: An Open Web-Scale Filtered Dataset of Interleaved Image-Text Documents”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine. 2023. URL: [http://papers.nips.cc/paper\\_files/paper/2023/hash/e2cfb719f58585f779d0a4f9f07bd618-Abstract-Datasets\\_and\\_Benchmarks.html](http://papers.nips.cc/paper_files/paper/2023/hash/e2cfb719f58585f779d0a4f9f07bd618-Abstract-Datasets_and_Benchmarks.html).
- [88] Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. “What matters when building vision-language models?” In: *CoRR* abs/2405.02246 (2024). DOI: 10.48550/ARXIV.2405.02246. arXiv: 2405.02246. URL: <https://doi.org/10.48550/arXiv.2405.02246>.
- [89] Chris Lewis and Jim Whitehead. “The whats and the whys of games and software engineering”. In: *Proceedings of the 1st international workshop on games and software engineering*. 2011, pp. 1–4.
- [90] Chris Lewis, Jim Whitehead, and Noah Wardrip-Fruin. “What went wrong: a taxonomy of video game bugs”. In: *International Conference on the Foundations of Digital Games, FDG '10, Pacific Grove, CA, USA, June 19-21, 2010*. Ed. by Ian Horswill and Yusuf Pisan. ACM, 2010, pp. 108–115. DOI: 10.1145/1822348.1822363. URL: <https://doi.org/10.1145/1822348.1822363>.
- [91] Bo Li, Hao Zhang, Kaichen Zhang, et al. *LLaVA-NeXT: What Else Influences Visual Instruction Tuning Beyond Data?* May 2024. URL: <https://llava-vl.github.io/blog/2024-05-25-llava-next-ablations/>.
- [92] Bo Li, Peiyuan Zhang, Jingkang Yang, Yuanhan Zhang, Fanyi Pu, and Ziwei Liu. “OtterHD: A High-Resolution Multi-modality Model”. In: *CoRR* abs/2311.04219 (2023). DOI: 10.48550/ARXIV.2311.04219. arXiv: 2311.04219. URL: <https://doi.org/10.48550/arXiv.2311.04219>.
- [93] Bohao Li, Rui Wang, Guangzhi Wang, Yuying Ge, Yixiao Ge, and Ying Shan. “SEED-Bench: Benchmarking Multimodal LLMs with Generative Comprehension”. In: *CoRR* abs/2307.16125 (2023). DOI: 10.48550/ARXIV.2307.16125. arXiv: 2307.16125. URL: <https://doi.org/10.48550/arXiv.2307.16125>.
- [94] Chunyuan Li. “Large Multimodal Models: Notes on CVPR 2023 Tutorial”. In: *CoRR* abs/2306.14895 (2023). DOI: 10.48550/ARXIV.2306.14895. arXiv: 2306.14895. URL: <https://doi.org/10.48550/arXiv.2306.14895>.
- [95] Chunyuan Li, Cliff Wong, Sheng Zhang, et al. “LLaVA-Med: Training a Large Language-and-Vision Assistant for Biomedicine in One Day”. In: *CoRR* abs/2306.00890 (2023). DOI: 10.48550/ARXIV.2306.00890. arXiv: 2306.00890. URL: <https://doi.org/10.48550/arXiv.2306.00890>.

- [96] Junnan Li, Dongxu Li, Silvio Savarese, and Steven C. H. Hoi. “BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models”. In: *CoRR* abs/2301.12597 (2023). DOI: 10.48550/ARXIV.2301.12597. arXiv: 2301.12597. URL: <https://doi.org/10.48550/arXiv.2301.12597>.
- [97] Junnan Li, Dongxu Li, Caiming Xiong, and Steven C. H. Hoi. “BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding and Generation”. In: *CoRR* abs/2201.12086 (2022). arXiv: 2201.12086. URL: <https://arxiv.org/abs/2201.12086>.
- [98] Raymond Li, Loubna Ben Allal, Yangtian Zi, et al. “StarCoder: may the source be with you!” In: *CoRR* abs/2305.06161 (2023). DOI: 10.48550/ARXIV.2305.06161. arXiv: 2305.06161. URL: <https://doi.org/10.48550/arXiv.2305.06161>.
- [99] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. “Evaluating Object Hallucination in Large Vision-Language Models”. In: *CoRR* abs/2305.10355 (2023). DOI: 10.48550/ARXIV.2305.10355. arXiv: 2305.10355. URL: <https://doi.org/10.48550/arXiv.2305.10355>.
- [100] Zhaoshuo Li, Thomas Müller, Alex Evans, et al. “Neuralangelo: High-Fidelity Neural Surface Reconstruction”. In: *CoRR* abs/2306.03092 (2023). DOI: 10.48550/ARXIV.2306.03092. arXiv: 2306.03092. URL: <https://doi.org/10.48550/arXiv.2306.03092>.
- [101] Bin Lin, Yang Ye, Bin Zhu, et al. “Video-LLaVA: Learning United Visual Representation by Alignment Before Projection”. In: *CoRR* abs/2311.10122 (2023). DOI: 10.48550/ARXIV.2311.10122. arXiv: 2311.10122. URL: <https://doi.org/10.48550/arXiv.2311.10122>.
- [102] Dayi Lin, Cor-Paul Bezemer, and Ahmed E. Hassan. “Identifying gameplay videos that exhibit bugs in computer games”. In: *Empir. Softw. Eng.* 24.6 (2019), pp. 4006–4033. DOI: 10.1007/S10664-019-09733-6. URL: <https://doi.org/10.1007/s10664-019-09733-6>.
- [103] Dayi Lin, Cor-Paul Bezemer, Ying Zou, and Ahmed E. Hassan. “An empirical study of game reviews on the Steam platform”. In: *Empir. Softw. Eng.* 24.1 (2019), pp. 170–207. DOI: 10.1007/S10664-018-9627-4. URL: <https://doi.org/10.1007/s10664-018-9627-4>.
- [104] Ziyi Lin, Chris Liu, Renrui Zhang, et al. “SPHINX: The Joint Mixing of Weights, Tasks, and Visual Embeddings for Multi-modal Large Language Models”. In: *CoRR* abs/2311.07575 (2023). DOI: 10.48550/ARXIV.2311.07575. arXiv: 2311.07575. URL: <https://doi.org/10.48550/arXiv.2311.07575>.
- [105] Carlos García Ling, Konrad Tollmar, and Linus Gisslén. “Using Deep Convolutional Neural Networks to Detect Rendered Glitches in Video Games”. In: *CoRR* abs/2406.08231 (2024). DOI: 10.48550/ARXIV.2406.08231. arXiv: 2406.08231. URL: <https://doi.org/10.48550/arXiv.2406.08231>.

- [106] Fuxiao Liu, Tianrui Guan, Zongxia Li, et al. “HallusionBench: You See What You Think? Or You Think What You See? An Image-Context Reasoning Benchmark Challenging for GPT-4V(ision), LLaVA-1.5, and Other Multimodality Models”. In: *CoRR* abs/2310.14566 (2023). DOI: 10.48550/ARXIV.2310.14566. arXiv: 2310.14566. URL: <https://doi.org/10.48550/arXiv.2310.14566>.
- [107] Fuxiao Liu, Kevin Lin, Linjie Li, Jianfeng Wang, Yaser Yacoob, and Lijuan Wang. “Aligning Large Multi-Modal Model with Robust Instruction Tuning”. In: *CoRR* abs/2306.14565 (2023). DOI: 10.48550/ARXIV.2306.14565. arXiv: 2306.14565. URL: <https://doi.org/10.48550/arXiv.2306.14565>.
- [108] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. “Improved Baselines with Visual Instruction Tuning”. In: *CoRR* abs/2310.03744 (2023). DOI: 10.48550/ARXIV.2310.03744. arXiv: 2310.03744. URL: <https://doi.org/10.48550/arXiv.2310.03744>.
- [109] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. “Improved Baselines with Visual Instruction Tuning”. In: *CoRR* abs/2310.03744 (2023). DOI: 10.48550/ARXIV.2310.03744. arXiv: 2310.03744. URL: <https://doi.org/10.48550/arXiv.2310.03744>.
- [110] Haotian Liu, Chunyuan Li, Yuheng Li, et al. *LLaVA-NeXT: Improved reasoning, OCR, and world knowledge*. Jan. 2024. URL: <https://llava-vl.github.io/blog/2024-01-30-llava-next/>.
- [111] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. “Visual Instruction Tuning”. In: *CoRR* abs/2304.08485 (2023). DOI: 10.48550/ARXIV.2304.08485. arXiv: 2304.08485. URL: <https://doi.org/10.48550/arXiv.2304.08485>.
- [112] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. “Visual instruction tuning”. In: *Advances in neural information processing systems* 36 (2024).
- [113] Shilong Liu, Hao Cheng, Haotian Liu, et al. “LLaVA-Plus: Learning to Use Tools for Creating Multimodal Agents”. In: *CoRR* abs/2311.05437 (2023). DOI: 10.48550/ARXIV.2311.05437. arXiv: 2311.05437. URL: <https://doi.org/10.48550/arXiv.2311.05437>.
- [114] Yuan Liu, Haodong Duan, Yuanhan Zhang, et al. “MMBench: Is Your Multimodal Model an All-around Player?” In: *CoRR* abs/2307.06281 (2023). DOI: 10.48550/ARXIV.2307.06281. arXiv: 2307.06281. URL: <https://doi.org/10.48550/arXiv.2307.06281>.
- [115] Pan Lu, Hritik Bansal, Tony Xia, et al. “MathVista: Evaluating Mathematical Reasoning of Foundation Models in Visual Contexts”. In: *arXiv preprint arXiv:2310.02255* (2023).
- [116] Pan Lu, Swaroop Mishra, Tony Xia, et al. “Learn to Explain: Multimodal Reasoning via Thought Chains for Science Question Answering”. In: *CoRR* abs/2209.09513 (2022). DOI: 10.48550/ARXIV.2209.09513. arXiv: 2209.09513. URL: <https://doi.org/10.48550/arXiv.2209.09513>.

- [117] Pan Lu, Liang Qiu, Jiaqi Chen, et al. “IconQA: A New Benchmark for Abstract Diagram Understanding and Visual Language Reasoning”. In: *CoRR* abs/2110.13214 (2021). arXiv: 2110.13214. URL: <https://arxiv.org/abs/2110.13214>.
- [118] Luma AI. *Luma Dream Machine*. <https://lumalabs.ai/>. Version X.X. 2024.
- [119] Huaishao Luo, Lei Ji, Ming Zhong, et al. “CLIP4Clip: An empirical study of CLIP for end to end video clip retrieval and captioning”. In: *Neurocomputing* 508 (2022), pp. 293–304. DOI: 10.1016/J.NEUCOM.2022.07.028. URL: <https://doi.org/10.1016/j.neucom.2022.07.028>.
- [120] Zijin Luo, Matthew Guzdial, and Mark Riedl. “Making CNNs for video parsing accessible: event extraction from dota2 gameplay video using transfer, zero-shot, and network pruning”. In: *Proceedings of the 14th International Conference on the Foundations of Digital Games*. 2019, pp. 1–10.
- [121] Weiyu Ma, Qirui Mi, Xue Yan, et al. “Large Language Models Play StarCraft II: Benchmarks and A Chain of Summarization Approach”. In: *CoRR* abs/2312.11865 (2023). DOI: 10.48550/ARXIV.2312.11865. arXiv: 2312.11865. URL: <https://doi.org/10.48550/arXiv.2312.11865>.
- [122] Finlay Macklon, Mohammad Reza Taesiri, Markos Viggiano, et al. “Automatically Detecting Visual Bugs in HTML5 Canvas Games”. In: *37th IEEE/ACM International Conference on Automated Software Engineering, ASE 2022, Rochester, MI, USA, October 10-14, 2022*. ACM, 2022, 15:1–15:11. DOI: 10.1145/3551349.3556913. URL: <https://doi.org/10.1145/3551349.3556913>.
- [123] Laura MacLeod, Margaret-Anne D. Storey, and Andreas Bergen. “Code, camera, action: how software developers document and share program knowledge using YouTube”. In: *Proceedings of the 2015 IEEE 23rd International Conference on Program Comprehension, ICPC 2015, Florence/Firenze, Italy, May 16-24, 2015*. Ed. by Andrea De Lucia, Christian Bird, and Rocco Oliveto. IEEE Computer Society, 2015, pp. 104–114. DOI: 10.1109/ICPC.2015.19. URL: <https://doi.org/10.1109/ICPC.2015.19>.
- [124] Rashi Maheshwari. *AI in Healthcare: What it is and how it works*. May 2023. URL: <https://www.forbes.com/advisor/in/health-insurance/ai-in-healthcare/>.
- [125] Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, and Sayak Paul. *PEFT: State-of-the-art Parameter-Efficient Fine-Tuning Methods*. <https://github.com/huggingface/peft>. 2022.
- [126] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. “OK-VQA: A Visual Question Answering Benchmark Requiring External Knowledge”. In: *CoRR* abs/1906.00067 (2019). arXiv: 1906.00067. URL: <http://arxiv.org/abs/1906.00067>.
- [127] Mashable. “You can use Microsoft’s Copilot AI chatbot to learn how to play ‘Minecraft’”. In: *Mashable* (2024). Accessed on June 20, 2024. URL: <https://mashable.com/article/microsoft-build-2024-copilot-minecraft>.

- [128] Joe McKendrick. *Artificial Intelligence Opens Up the World of Financial Services*. Feb. 2023. URL: <https://www.forbes.com/sites/joemckendrick/2023/02/24/artificial-intelligence-opens-up-the-world-of-financial-services/>.
- [129] Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, et al. “MM1: Methods, Analysis & Insights from Multimodal LLM Pre-training”. In: *CoRR* abs/2403.09611 (2024). DOI: 10.48550/ARXIV.2403.09611. arXiv: 2403.09611. URL: <https://doi.org/10.48550/arXiv.2403.09611>.
- [130] AI at Meta. “Llama 3 Model Card”. In: (2024). URL: [https://github.com/meta-llama/llama3/blob/main/MODEL\\_CARD.md](https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md).
- [131] Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. “OCR-VQA: Visual Question Answering by Reading Text in Images”. In: *2019 International Conference on Document Analysis and Recognition, ICDAR 2019, Sydney, Australia, September 20-25, 2019*. IEEE, 2019, pp. 947–952. DOI: 10.1109/ICDAR.2019.00156. URL: <https://doi.org/10.1109/ICDAR.2019.00156>.
- [132] Emerson R. Murphy-Hill, Thomas Zimmermann, and Nachiappan Nagappan. “Cowboys, ankle sprains, and keepers of quality: how is video game development different from software development?” In: *36th International Conference on Software Engineering, ICSE '14, Hyderabad, India - May 31 - June 07, 2014*. Ed. by Pankaj Jalote, Lionel C. Briand, and André van der Hoek. ACM, 2014, pp. 1–11. DOI: 10.1145/2568225.2568226. URL: <https://doi.org/10.1145/2568225.2568226>.
- [133] Alfredo Nantes, Ross Brown, and Frédéric Maire. “A Framework for the Semi-Automatic Testing of Video Games”. In: *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference, October 22-24, 2008, Stanford, California, USA*. Ed. by Christian Darken and Michael Mateas. The AAAI Press, 2008. URL: <http://www.aaai.org/Library/AIIDE/2008/aiide08-033.php>.
- [134] Bolin Ni, Houwen Peng, Minghao Chen, et al. “Expanding Language-Image Pretrained Models for General Video Recognition”. In: *CoRR* abs/2208.02816 (2022). DOI: 10.48550/ARXIV.2208.02816. arXiv: 2208.02816. URL: <https://doi.org/10.48550/arXiv.2208.02816>.
- [135] NVIDIA ACE for Games Sparks Life into Virtual Characters with Generative AI. June 2023. URL: <https://nvidianews.nvidia.com/news/nvidia-ace-for-games-sparks-life-into-virtual-characters-with-generative-ai>.
- [136] OpenAI. *GPT-4o Model Card*. <https://openai.com/index/hello-gpt-4o/>. Accessed: 2024-07-07. 2024.
- [137] *OpenAI’s GPT-4V*. 2023. URL: <https://chat.openai.com/>.
- [138] *OpenAI’s GPT-4V(ision)*. 2023. URL: <https://chat.openai.com/>.

- [139] Long Ouyang, Jeff Wu, Xu Jiang, et al. “Training language models to follow instructions with human feedback”. In: *CoRR* abs/2203.02155 (2022). DOI: 10.48550/ARXIV.2203.02155. arXiv: 2203.02155. URL: <https://doi.org/10.48550/arXiv.2203.02155>.
- [140] Joon Sung Park, Joseph C. O’Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. “Generative Agents: Interactive Simulacra of Human Behavior”. In: *CoRR* abs/2304.03442 (2023). DOI: 10.48550/ARXIV.2304.03442. arXiv: 2304.03442. URL: <https://doi.org/10.48550/arXiv.2304.03442>.
- [141] Luca Pascarella, Fabio Palomba, Massimiliano Di Penta, and Alberto Bacchelli. “How is video game development different from software development in open source?” In: *Proceedings of the 15th International Conference on Mining Software Repositories, MSR 2018, Gothenburg, Sweden, May 28-29, 2018*. Ed. by Andy Zaidman, Yasutaka Kamei, and Emily Hill. ACM, 2018, pp. 392–402. DOI: 10.1145/3196398.3196418. URL: <https://doi.org/10.1145/3196398.3196418>.
- [142] Zhiliang Peng, Wenhui Wang, Li Dong, et al. “Kosmos-2: Grounding Multimodal Large Language Models to the World”. In: *CoRR* abs/2306.14824 (2023). DOI: 10.48550/ARXIV.2306.14824. arXiv: 2306.14824. URL: <https://doi.org/10.48550/arXiv.2306.14824>.
- [143] Fábio Petrillo, Marcelo Soares Pimenta, Francisco M. Trindade, and Carlos Dietrich. “What went wrong? A survey of problems in game development”. In: *Comput. Entertain.* 7.1 (2009), 13:1–13:22. DOI: 10.1145/1486508.1486521. URL: <https://doi.org/10.1145/1486508.1486521>.
- [144] Johannes Pfau, Antonios Liapis, Georg Volkmar, Georgios N. Yannakakis, and Rainer Malaka. “Dungeons & Replicants: Automated Game Balancing via Deep Player Behavior Modeling”. In: *IEEE Conference on Games, CoG 2020, Osaka, Japan, August 24-27, 2020*. IEEE, 2020, pp. 431–438. DOI: 10.1109/COG47356.2020.9231958. URL: <https://doi.org/10.1109/CoG47356.2020.9231958>.
- [145] Bryan A. Plummer, Liwei Wang, Chris M. Cervantes, Juan C. Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. “Flickr30k Entities: Collecting Region-to-Phrase Correspondences for Richer Image-to-Sentence Models”. In: *CoRR* abs/1505.04870 (2015). arXiv: 1505.04870. URL: <http://arxiv.org/abs/1505.04870>.
- [146] Cristiano Politowski, Fábio Petrillo, and Yann-Gaël Guéhéneuc. “A Survey of Video Game Testing”. In: *CoRR* abs/2103.06431 (2021). arXiv: 2103.06431. URL: <https://arxiv.org/abs/2103.06431>.
- [147] Cristiano Politowski, Fábio Petrillo, Gabriel Cavalheiro Ullmann, Josias de Andrade Werly, and Yann-Gaël Guéhéneuc. “Dataset of Video Game Development Problems”. In: *CoRR* abs/2001.00491 (2020). arXiv: 2001.00491. URL: <http://arxiv.org/abs/2001.00491>.

- [148] Luca Ponzanelli, Gabriele Bavota, Andrea Mocci, et al. “Too long; didn’t watch!: extracting relevant fragments from software development video tutorials”. In: *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*. Ed. by Laura K. Dillon, Willem Visser, and Laurie A. Williams. ACM, 2016, pp. 261–272. DOI: 10.1145/2884781.2884824. URL: <https://doi.org/10.1145/2884781.2884824>.
- [149] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. “DreamFusion: Text-to-3D using 2D Diffusion”. In: *CoRR* abs/2209.14988 (2022). DOI: 10.48550/ARXIV.2209.14988. arXiv: 2209.14988. URL: <https://doi.org/10.48550/arXiv.2209.14988>.
- [150] Haotong Qin, Ge-Peng Ji, Salman H. Khan, Deng-Ping Fan, Fahad Shahbaz Khan, and Luc Van Gool. “How Good is Google Bard’s Visual Understanding? An Empirical Study on Open Challenges”. In: *CoRR* abs/2307.15016 (2023). DOI: 10.48550/ARXIV.2307.15016. arXiv: 2307.15016. URL: <https://doi.org/10.48550/arXiv.2307.15016>.
- [151] Alec Radford, Jong Wook Kim, Chris Hallacy, et al. “Learning Transferable Visual Models From Natural Language Supervision”. In: *CoRR* abs/2103.00020 (2021). arXiv: 2103.00020. URL: <https://arxiv.org/abs/2103.00020>.
- [152] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. “Language models are unsupervised multitask learners”. In: *OpenAI blog* 1.8 (2019), p. 9.
- [153] Colin Raffel, Noam Shazeer, Adam Roberts, et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *CoRR* abs/1910.10683 (2019). arXiv: 1910.10683. URL: <http://arxiv.org/abs/1910.10683>.
- [154] Farrukh Rahman. “Weak Supervision for Label Efficient Visual Bug Detection”. In: *CoRR* abs/2309.11077 (2023). DOI: 10.48550/ARXIV.2309.11077. arXiv: 2309.11077. URL: <https://doi.org/10.48550/arXiv.2309.11077>.
- [155] Geeta Rani, Upasana Pandey, Aniket Anil Wagde, and Vijaypal Singh Dhaka. “A deep reinforcement learning technique for bug detection in video games”. In: *International Journal of Information Technology* 15.1 (2023), pp. 355–367.
- [156] Scott E. Reed, Konrad Zolna, Emilio Parisotto, et al. “A Generalist Agent”. In: *CoRR* abs/2205.06175 (2022). DOI: 10.48550/ARXIV.2205.06175. arXiv: 2205.06175. URL: <https://doi.org/10.48550/arXiv.2205.06175>.
- [157] *Report: Epic Games Business Breakdown & Founding Story*. <https://research.contrary.com/reports/epic-games>. (Accessed on 11/15/2023).
- [158] Grand View Research. “Video Game Market Size, Share And Growth Report, 2030”. In: *Grand View Research* (2023). URL: <https://www.grandviewresearch.com/industry-analysis/video-game-market>.

- [159] Anna Rohrbach, Lisa Anne Hendricks, Kaylee Burns, Trevor Darrell, and Kate Saenko. “Object Hallucination in Image Captioning”. In: *CoRR* abs/1809.02156 (2018). arXiv: 1809.02156. URL: <http://arxiv.org/abs/1809.02156>.
- [160] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. “High-Resolution Image Synthesis with Latent Diffusion Models”. In: *CoRR* abs/2112.10752 (2021). arXiv: 2112.10752. URL: <https://arxiv.org/abs/2112.10752>.
- [161] Shaghayegh Roohi, Christian Guckelsberger, Asko Relas, Henri Heiskanen, Jari Takatalo, and Perttu Hämäläinen. “Predicting Game Difficulty and Engagement Using AI Players”. In: *Proc. ACM Hum. Comput. Interact.* 5.CHI PLAY (2021), 231:1–231:17. DOI: 10.1145/3474658. URL: <https://doi.org/10.1145/3474658>.
- [162] Olga Russakovsky, Jia Deng, Hao Su, et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *CoRR* abs/1409.0575 (2014). arXiv: 1409.0575. URL: <http://arxiv.org/abs/1409.0575>.
- [163] Ronnie ES Santos, Cleyton VC Magalhães, Luiz Fernando Capretz, Jorge S Correia-Neto, Fabio QB da Silva, and Abdelrahman Saher. “Computer games are serious business and so is their quality: particularities of software testing in game development from the perspective of practitioners”. In: *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 2018, pp. 1–10.
- [164] Christoph Schuhmann, Romain Beaumont, Richard Vencu, et al. “LAION-5B: An open large-scale dataset for training next generation image-text models”. In: *CoRR* abs/2210.08402 (2022). DOI: 10.48550/ARXIV.2210.08402. arXiv: 2210.08402. URL: <https://doi.org/10.48550/arXiv.2210.08402>.
- [165] Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. “A-OKVQA: A Benchmark for Visual Question Answering using World Knowledge”. In: *CoRR* abs/2206.01718 (2022). DOI: 10.48550/ARXIV.2206.01718. arXiv: 2206.01718. URL: <https://doi.org/10.48550/arXiv.2206.01718>.
- [166] Xiao Shao, Weifu Jiang, Fei Zuo, and Mengqing Liu. “SwarmBrain: Embodied agent for real-time strategy game StarCraft II via large language models”. In: *CoRR* abs/2401.17749 (2024). DOI: 10.48550/ARXIV.2401.17749. arXiv: 2401.17749. URL: <https://doi.org/10.48550/arXiv.2401.17749>.
- [167] Baifeng Shi, Ziyang Wu, Maolin Mao, Xin Wang, and Trevor Darrell. “When Do We Not Need Larger Vision Models?” In: *CoRR* abs/2403.13043 (2024). DOI: 10.48550/ARXIV.2403.13043. arXiv: 2403.13043. URL: <https://doi.org/10.48550/arXiv.2403.13043>.
- [168] Noah Shinn, Beck Labash, and Ashwin Gopinath. “Reflexion: an autonomous agent with dynamic memory and self-reflection”. In: *CoRR* abs/2303.11366 (2023). DOI: 10.48550/ARXIV.2303.11366. arXiv: 2303.11366. URL: <https://doi.org/10.48550/arXiv.2303.11366>.

- [169] Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. “TextCaps: a Dataset for Image Captioning with Reading Comprehension”. In: *CoRR* abs/2003.12462 (2020). arXiv: 2003.12462. URL: <https://arxiv.org/abs/2003.12462>.
- [170] Amanpreet Singh, Vivek Natarajan, Meet Shah, et al. “Towards VQA Models that can Read”. In: *CoRR* abs/1904.08920 (2019). arXiv: 1904.08920. URL: <http://arxiv.org/abs/1904.08920>.
- [171] Konstantinos Souchleris, George K Sidiropoulos, and George A Papakostas. “Reinforcement Learning in Game Industry—Review, Prospects and Challenges”. In: *Applied Sciences* 13.4 (2023), p. 2443.
- [172] Patrick Stacey and Joe Nandhakumar. “A temporal perspective of the computer game development process”. In: *Inf. Syst. J.* 19.5 (2009), pp. 479–497. DOI: 10.1111/J.1365-2575.2007.00273.X. URL: <https://doi.org/10.1111/j.1365-2575.2007.00273.x>.
- [173] Mohammad Reza Taesiri and Cor-Paul Bezemer. “VideoGameBunny: Towards vision assistants for video games”. In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2025.
- [174] Mohammad Reza Taesiri, Tianjun Feng, Cor-Paul Bezemer, and Anh Nguyen. “GlitchBench: Can large multimodal models detect video game glitches?” In: *CoRR* abs/2312.05291 (2023). DOI: 10.48550/ARXIV.2312.05291. arXiv: 2312.05291. URL: <https://doi.org/10.48550/arXiv.2312.05291>.
- [175] Mohammad Reza Taesiri, Moslem Habibi, and Mohammad Amin Fazli. “A Video Game Testing Method Utilizing Deep Learning”. In: *Iran Journal of Computer Science* 17.2 (2020), pp. 26–33.
- [176] Mohammad Reza Taesiri, Finlay Macklon, and Cor-Paul Bezemer. “CLIP meets GamePhysics: Towards bug identification in gameplay videos using zero-shot transfer learning”. In: *CoRR* abs/2203.11096 (2022). DOI: 10.48550/ARXIV.2203.11096. arXiv: 2203.11096. URL: <https://doi.org/10.48550/arXiv.2203.11096>.
- [177] Mohammad Reza Taesiri, Finlay Macklon, Sarra Habchi, and Cor-Paul Bezemer. “Searching bug instances in gameplay video repositories”. In: *IEEE Transactions on Games* (2024).
- [178] 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”. In: *CoRR* abs/2210.02506 (2022). DOI: 10.48550/ARXIV.2210.02506. arXiv: 2210.02506. URL: <https://doi.org/10.48550/arXiv.2210.02506>.

- [179] Mohammad Reza Taesiri, Giang Nguyen, Sarra Habchi, Cor-Paul Bezemer, and Anh Nguyen. “ImageNet-Hard: The Hardest Images Remaining from a Study of the Power of Zoom and Spatial Biases in Image Classification”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine. 2023. URL: [http://papers.nips.cc/paper\\_files/paper/2023/hash/706390d6f9208b03bc54f97ac3cfe99e-Abstract-Datasets\\_and\\_Benchmarks.html](http://papers.nips.cc/paper_files/paper/2023/hash/706390d6f9208b03bc54f97ac3cfe99e-Abstract-Datasets_and_Benchmarks.html).
- [180] Weihao Tan, Ziluo Ding, Wentao Zhang, et al. “Towards General Computer Control: A Multimodal Agent for Red Dead Redemption II as a Case Study”. In: *CoRR* abs/2403.03186 (2024). DOI: 10.48550/ARXIV.2403.03186. arXiv: 2403.03186. URL: <https://doi.org/10.48550/arXiv.2403.03186>.
- [181] Chameleon Team. “Chameleon: Mixed-Modal Early-Fusion Foundation Models”. In: *CoRR* abs/2405.09818 (2024). DOI: 10.48550/ARXIV.2405.09818. arXiv: 2405.09818. URL: <https://doi.org/10.48550/arXiv.2405.09818>.
- [182] Gemini Team, Rohan Anil, Sebastian Borgeaud, et al. “Gemini: a family of highly capable multimodal models”. In: *arXiv preprint arXiv:2312.11805* (2023).
- [183] *The Skyrim Bee Bug*. Reddit. During development, the cart in Skyrim’s intro was affected by a collision issue with an in-game bee. 2021. URL: [https://www.reddit.com/r/xboxone/comments/p6pxzk/during\\_development\\_the\\_cart\\_in\\_skyrims\\_intro\\_was/](https://www.reddit.com/r/xboxone/comments/p6pxzk/during_development_the_cart_in_skyrims_intro_was/).
- [184] Tristan Thrush, Ryan Jiang, Max Bartolo, et al. “Winoground: Probing Vision and Language Models for Visio-Linguistic Compositionality”. In: *CoRR* abs/2204.03162 (2022). DOI: 10.48550/ARXIV.2204.03162. arXiv: 2204.03162. URL: <https://doi.org/10.48550/arXiv.2204.03162>.
- [185] Shengbang Tong, Ellis Brown, Penghao Wu, et al. “Cambrian-1: A Fully Open, Vision-Centric Exploration of Multimodal LLMs”. In: *arXiv preprint arXiv:2406.16860* (2024).
- [186] Hugo Touvron, Louis Martin, Kevin Stone, et al. “Llama 2: Open Foundation and Fine-Tuned Chat Models”. In: *CoRR* abs/2307.09288 (2023). DOI: 10.48550/ARXIV.2307.09288. arXiv: 2307.09288. URL: <https://doi.org/10.48550/arXiv.2307.09288>.
- [187] Chintan Trivedi, Antonios Liapis, and Georgios N. Yannakakis. “Contrastive Learning of Generalized Game Representations”. In: *CoRR* abs/2106.10060 (2021). arXiv: 2106.10060. URL: <https://arxiv.org/abs/2106.10060>.
- [188] Andrew Truelove, Eduardo Santana de Almeida, and Iftekhar Ahmed. “We’ll Fix It in Post: What Do Bug Fixes in Video Game Update Notes Tell Us?” In: *CoRR* abs/2103.03997 (2021). arXiv: 2103.03997. URL: <https://arxiv.org/abs/2103.03997>.

- [189] J. Tuovenen, Mourad Oussalah, and Panos Kostakos. “MAuto: Automatic Mobile Game Testing Tool Using Image-Matching Based Approach”. In: *Comput. Games J.* 8.3-4 (2019), pp. 215–239. DOI: 10.1007/S40869-019-00087-Z. URL: <https://doi.org/10.1007/s40869-019-00087-z>.
- [190] Simon Varvaressos, Kim Lavoie, Sébastien Gaboury, and Sylvain Hallé. “Automated Bug Finding in Video Games: A Case Study for Runtime Monitoring”. In: *Comput. Entertain.* 15.1 (2017), 1:1–1:28. DOI: 10.1145/2700529. URL: <https://doi.org/10.1145/2700529>.
- [191] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. “Attention Is All You Need”. In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: <http://arxiv.org/abs/1706.03762>.
- [192] Markos Viggiano, Dale Paas, Chris Buzon, and Cor-Paul Bezemer. “Identifying Similar Test Cases That Are Specified in Natural Language”. In: *CoRR* abs/2110.07733 (2021). arXiv: 2110.07733. URL: <https://arxiv.org/abs/2110.07733>.
- [193] Markos Viggiano, Dale Paas, Chris Buzon, and Cor-Paul Bezemer. “Using Natural Language Processing Techniques to Improve Manual Test Case Descriptions”. In: *44th IEEE/ACM International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP) 2022, Pittsburgh, PA, USA, May 22-24, 2022*. IEEE, 2022, pp. 311–320. DOI: 10.1109/ICSE-SEIP55303.2022.9794054. URL: <https://doi.org/10.1109/ICSE-SEIP55303.2022.9794054>.
- [194] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, et al. “Grandmaster level in StarCraft II using multi-agent reinforcement learning”. In: *Nat.* 575.7782 (2019), pp. 350–354. DOI: 10.1038/S41586-019-1724-Z. URL: <https://doi.org/10.1038/s41586-019-1724-z>.
- [195] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, et al. “Voyager: An Open-Ended Embodied Agent with Large Language Models”. In: *CoRR* abs/2305.16291 (2023). DOI: 10.48550/ARXIV.2305.16291. arXiv: 2305.16291. URL: <https://doi.org/10.48550/arXiv.2305.16291>.
- [196] Yue Wang, Weishi Wang, Shafiq R. Joty, and Steven C. H. Hoi. “CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation”. In: *CoRR* abs/2109.00859 (2021). arXiv: 2109.00859. URL: <https://arxiv.org/abs/2109.00859>.
- [197] Zihao Wang, Shaofei Cai, Anji Liu, Xiaojian Ma, and Yitao Liang. “Describe, Explain, Plan and Select: Interactive Planning with Large Language Models Enables Open-World Multi-Task Agents”. In: *CoRR* abs/2302.01560 (2023). DOI: 10.48550/ARXIV.2302.01560. arXiv: 2302.01560. URL: <https://doi.org/10.48550/arXiv.2302.01560>.
- [198] Zihao Wang, Shaofei Cai, Anji Liu, et al. “JARVIS-1: Open-World Multi-task Agents with Memory-Augmented Multimodal Language Models”. In: *CoRR* abs/2311.05997 (2023). DOI: 10.48550/ARXIV.2311.05997. arXiv: 2311.05997. URL: <https://doi.org/10.48550/arXiv.2311.05997>.

- [199] Jason Wei, Xuezhi Wang, Dale Schuurmans, et al. “Chain of Thought Prompting Elicits Reasoning in Large Language Models”. In: *CoRR* abs/2201.11903 (2022). arXiv: 2201.11903. URL: <https://arxiv.org/abs/2201.11903>.
- [200] Licheng Wen, Xuemeng Yang, Daocheng Fu, et al. “On the Road with GPT-4V(ision): Early Explorations of Visual-Language Model on Autonomous Driving”. In: *CoRR* abs/2311.05332 (2023). DOI: 10.48550/ARXIV.2311.05332. arXiv: 2311.05332. URL: <https://doi.org/10.48550/arXiv.2311.05332>.
- [201] Benedict Wilkins and Kostas Stathis. “Learning to Identify Perceptual Bugs in 3D Video Games”. In: *CoRR* abs/2202.12884 (2022). arXiv: 2202.12884. URL: <https://arxiv.org/abs/2202.12884>.
- [202] Thomas Wolf, Lysandre Debut, Victor Sanh, et al. “HuggingFace’s Transformers: State-of-the-art Natural Language Processing”. In: *CoRR* abs/1910.03771 (2019). arXiv: 1910.03771. URL: <http://arxiv.org/abs/1910.03771>.
- [203] Mitchell Wortsman, Gabriel Ilharco, Mike Li, et al. “Robust fine-tuning of zero-shot models”. In: *CoRR* abs/2109.01903 (2021). arXiv: 2109.01903. URL: <https://arxiv.org/abs/2109.01903>.
- [204] Yang Wu, Shilong Wang, Hao Yang, et al. “An Early Evaluation of GPT-4V(ision)”. In: *CoRR* abs/2310.16534 (2023). DOI: 10.48550/ARXIV.2310.16534. arXiv: 2310.16534. URL: <https://doi.org/10.48550/arXiv.2310.16534>.
- [205] Yue Wu, So Yeon Min, Shrimai Prabhunoye, et al. “SPRING: Studying Papers and Reasoning to play Games”. In: *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*. Ed. by Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine. 2023. URL: [http://papers.nips.cc/paper%5C\\_files/paper/2023/hash/46c2a9a6f2b2be68682013eb1173c801-Abstract-Conference.html](http://papers.nips.cc/paper%5C_files/paper/2023/hash/46c2a9a6f2b2be68682013eb1173c801-Abstract-Conference.html).
- [206] Bin Xiao, Haiping Wu, Weijian Xu, et al. “Florence-2: Advancing a Unified Representation for a Variety of Vision Tasks”. In: *CoRR* abs/2311.06242 (2023). DOI: 10.48550/ARXIV.2311.06242. arXiv: 2311.06242. URL: <https://doi.org/10.48550/arXiv.2311.06242>.
- [207] Xinrun Xu, Yuxin Wang, Chaoyi Xu, et al. “A Survey on Game Playing Agents and Large Models: Methods, Applications, and Challenges”. In: *CoRR* abs/2403.10249 (2024). DOI: 10.48550/ARXIV.2403.10249. arXiv: 2403.10249. URL: <https://doi.org/10.48550/arXiv.2403.10249>.
- [208] Zhiling Yan, Kai Zhang, Rong Zhou, Lifang He, Xiang Li, and Lichao Sun. “Multimodal ChatGPT for Medical Applications: an Experimental Study of GPT-4V”. In: *CoRR* abs/2310.19061 (2023). DOI: 10.48550/ARXIV.2310.19061. arXiv: 2310.19061. URL: <https://doi.org/10.48550/arXiv.2310.19061>.
- [209] Jianwei Yang, Hao Zhang, Feng Li, Xueyan Zou, Chunyuan Li, and Jianfeng Gao. “Set-of-Mark Prompting Unleashes Extraordinary Visual Grounding in GPT-4V”. In: *CoRR* abs/2310.11441 (2023). DOI: 10.48550/ARXIV.2310.11441. arXiv: 2310.11441. URL: <https://doi.org/10.48550/arXiv.2310.11441>.

- [210] Zhuolin Yang, Wei Ping, Zihan Liu, et al. “Re-ViLM: Retrieval-Augmented Visual Language Model for Zero and Few-Shot Image Captioning”. In: *CoRR* abs/2302.04858 (2023). DOI: 10.48550/ARXIV.2302.04858. arXiv: 2302.04858. URL: <https://doi.org/10.48550/arXiv.2302.04858>.
- [211] Qinghao Ye, Haiyang Xu, Guohai Xu, et al. “mPLUG-Owl: Modularization Empowers Large Language Models with Multimodality”. In: *CoRR* abs/2304.14178 (2023). DOI: 10.48550/ARXIV.2304.14178. arXiv: 2304.14178. URL: <https://doi.org/10.48550/arXiv.2304.14178>.
- [212] Zhenfei Yin, Jiong Wang, Jianjian Cao, et al. “LAMM: Language-Assisted Multi-Modal Instruction-Tuning Dataset, Framework, and Benchmark”. In: *CoRR* abs/2306.06687 (2023). DOI: 10.48550/ARXIV.2306.06687. arXiv: 2306.06687. URL: <https://doi.org/10.48550/arXiv.2306.06687>.
- [213] Weihao Yu, Zhengyuan Yang, Linjie Li, et al. “MM-Vet: Evaluating Large Multimodal Models for Integrated Capabilities”. In: *CoRR* abs/2308.02490 (2023). DOI: 10.48550/ARXIV.2308.02490. arXiv: 2308.02490. URL: <https://doi.org/10.48550/arXiv.2308.02490>.
- [214] Xiang Yue, Yuansheng Ni, Kai Zhang, et al. “MMM-U: A Massive Multi-discipline Multimodal Understanding and Reasoning Benchmark for Expert AGI”. In: *CoRR* abs/2311.16502 (2023). DOI: 10.48550/ARXIV.2311.16502. arXiv: 2311.16502. URL: <https://doi.org/10.48550/arXiv.2311.16502>.
- [215] Mehrshad Zandigohar, Mo Han, Deniz Erdogmus, and Gunar Schirner. “Towards Creating a Deployable Grasp Type Probability Estimator for a Prosthetic Hand”. In: *CoRR* abs/2101.05357 (2021). arXiv: 2101.05357. URL: <https://arxiv.org/abs/2101.05357>.
- [216] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. “From Recognition to Cognition: Visual Commonsense Reasoning”. In: *CoRR* abs/1811.10830 (2018). arXiv: 1811.10830. URL: <http://arxiv.org/abs/1811.10830>.
- [217] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. “Sigmoid Loss for Language Image Pre-Training”. In: *CoRR* abs/2303.15343 (2023). DOI: 10.48550/ARXIV.2303.15343. arXiv: 2303.15343. URL: <https://doi.org/10.48550/arXiv.2303.15343>.
- [218] Xiaohua Zhai, Xiao Wang, Basil Mustafa, et al. “LiT: Zero-Shot Transfer with Locked-image Text Tuning”. In: *CoRR* abs/2111.07991 (2021). arXiv: 2111.07991. URL: <https://arxiv.org/abs/2111.07991>.
- [219] JIEM ZHANG, MARK HARMAN, and MENG WANG. “An Empirical Study of the Non-determinism of ChatGPT in Code Generation”. In: ().
- [220] Longwen Zhang, Ziyu Wang, Qixuan Zhang, et al. “CLAY: A Controllable Large-scale Generative Model for Creating High-quality 3D Assets”. In: *CoRR* abs/2406.13897 (2024). DOI: 10.48550/ARXIV.2406.13897. arXiv: 2406.13897. URL: <https://doi.org/10.48550/arXiv.2406.13897>.

- [221] Marvin Zhang, Sergey Levine, and Chelsea Finn. “MEMO: Test Time Robustness via Adaptation and Augmentation”. In: *CoRR* abs/2110.09506 (2021). arXiv: 2110.09506. URL: <https://arxiv.org/abs/2110.09506>.
- [222] Shengyu Zhang, Linfeng Dong, Xiaoya Li, et al. “Instruction Tuning for Large Language Models: A Survey”. In: *CoRR* abs/2308.10792 (2023). DOI: 10.48550/ARXIV.2308.10792. arXiv: 2308.10792. URL: <https://doi.org/10.48550/arXiv.2308.10792>.
- [223] Wanpeng Zhang and Zongqing Lu. “AdaRefiner: Refining Decisions of Language Models with Adaptive Feedback”. In: *Findings of the Association for Computational Linguistics: NAACL 2024*. Association for Computational Linguistics, June 2024, pp. 782–799. URL: <https://aclanthology.org/2024.findings-naacl.50>.
- [224] Xiaoxuan Zhang and Adam M. Smith. “Retrieving videogame moments with natural language queries”. In: *Proceedings of the 14th International Conference on the Foundations of Digital Games, FDG 2019, San Luis Obispo, California, USA, August 26-30, 2019*. Ed. by Sebastian Deterding, Foad Khosmood, Johanna Pirker, and Thomas H. Apperley. ACM, 2019, 74:1–74:7. DOI: 10.1145/3337722.3341867. URL: <https://doi.org/10.1145/3337722.3341867>.
- [225] Ming Zhao, Peter Anderson, Vihan Jain, et al. “On the Evaluation of Vision-and-Language Navigation Instructions”. In: *CoRR* abs/2101.10504 (2021). arXiv: 2101.10504. URL: <https://arxiv.org/abs/2101.10504>.
- [226] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, et al. “Judging LLM-as-a-judge with MT-Bench and Chatbot Arena”. In: *CoRR* abs/2306.05685 (2023). DOI: 10.48550/ARXIV.2306.05685. arXiv: 2306.05685. URL: <https://doi.org/10.48550/arXiv.2306.05685>.
- [227] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, et al. “Judging LLM-as-a-judge with MT-Bench and Chatbot Arena”. In: *CoRR* abs/2306.05685 (2023). DOI: 10.48550/ARXIV.2306.05685. arXiv: 2306.05685. URL: <https://doi.org/10.48550/arXiv.2306.05685>.
- [228] Sipeng Zheng, Jiazheng Liu, Yicheng Feng, and Zongqing Lu. “Steve-Eye: Equipping LLM-based Embodied Agents with Visual Perception in Open Worlds”. In: *CoRR* abs/2310.13255 (2023). DOI: 10.48550/ARXIV.2310.13255. arXiv: 2310.13255. URL: <https://doi.org/10.48550/arXiv.2310.13255>.
- [229] Yan Zheng, Changjie Fan, Xiaofei Xie, et al. “Wuji: Automatic Online Combat Game Testing Using Evolutionary Deep Reinforcement Learning”. In: *34th IEEE/ACM International Conference on Automated Software Engineering, ASE 2019, San Diego, CA, USA, November 11-15, 2019*. IEEE, 2019, pp. 772–784. DOI: 10.1109/ASE.2019.00077. URL: <https://doi.org/10.1109/ASE.2019.00077>.

- [230] Kankan Zhou, Eason Lai, Wei Bin Au Yeong, Kyriakos Mouratidis, and Jing Jiang. “ROME: Evaluating Pre-trained Vision-Language Models on Reasoning beyond Visual Common Sense”. In: *CoRR* abs/2310.19301 (2023). DOI: 10.48550/ARXIV.2310.19301. arXiv: 2310.19301. URL: <https://doi.org/10.48550/arXiv.2310.19301>.
- [231] Yiyang Zhou, Chenhang Cui, Jaehong Yoon, et al. “Analyzing and Mitigating Object Hallucination in Large Vision-Language Models”. In: *CoRR* abs/2310.00754 (2023). DOI: 10.48550/ARXIV.2310.00754. arXiv: 2310.00754. URL: <https://doi.org/10.48550/arXiv.2310.00754>.
- [232] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. “MiniGPT-4: Enhancing Vision-Language Understanding with Advanced Large Language Models”. In: *CoRR* abs/2304.10592 (2023). DOI: 10.48550/ARXIV.2304.10592. arXiv: 2304.10592. URL: <https://doi.org/10.48550/arXiv.2304.10592>.
- [233] Yichen Zhu, Minjie Zhu, Ning Liu, Zhicai Ou, Xiaofeng Mou, and Jian Tang. “LLaVA-Phi: Efficient Multi-Modal Assistant with Small Language Model”. In: *CoRR* abs/2401.02330 (2024). DOI: 10.48550/ARXIV.2401.02330. arXiv: 2401.02330. URL: <https://doi.org/10.48550/arXiv.2401.02330>.

# Appendix A: Appendix GlitchBench

## For Can large multimodal models detect video game glitches?

### A.1 Implementation details

#### A.1.1 Details about model inference

For each open source model, we used the provided sample code and demo from their respective repositories. Minor modifications were made to enable automatic processing of all images with designated prompts. The results were then stored in individual CSV files for each model. For OtterHD, which offers an API, we used the API to submit each image along with the appropriate prompt and recorded the responses. Our experiment was done prior to the official release of the GPT-4V API, and we used the ChatGPT web version for the benchmark, using a Chrome extension to assist in the process.

We kept the temperature and other parameters of each model unchanged. The only modification involved increasing the `max_token` limit, ensuring that the model’s response length was not restricted.

#### A.1.2 Details about the judge

In our experiment, the Llama-2-70B model served as the judge. We utilized the API from perplexity.ai, which is compatible with OpenAI’s Python package. Additionally, we employed a custom system message, as detailed below:

*Your task is to compare a model-generated text with a ground truth reference, assessing whether the key information and themes are similarly conveyed, even if worded differently. Focus on semantic content, thematic alignment, and intent, rather than exact phrasing or word usage. Recognize synonyms, paraphrases, and different stylistic expressions as valid, provided they faithfully represent the ground truth's meaning. Offer feedback on the correlation between the texts and suggest improvements for alignment, while appreciating creative or varied linguistic expression that maintains the essence of the ground truth.*

*First analyze, then report the final answer in either of Yes or No*

## A.2 Additional results

### A.2.1 Breakdown of performance by various glitch types

Table A.1: Breakdown of Performance for Different LMMs by Various Glitch Types (%)

	Camera, User Interface, Lighting	Animation, Pose	Physics, Collision, Spawn	Rendering, Texture	Average Performance
GPT-4V	46.2	30.3	45.9	41.0	40.8
LLaVA-1.5-13B	36.5	27.0	35.7	43.3	35.6
LLaVA-1.5-7B	40.4	26.3	29.2	31.3	31.8
Long-SPHINX	30.8	20.4	29.7	23.9	26.2
SPHINX	25.0	17.8	26.1	26.9	23.9
OtterHD	15.4	19.1	25.9	20.9	20.3
InstructBLIP-7B	23.1	15.8	20.8	14.9	18.6
MiniGPT-v2	15.4	17.8	19.9	11.2	16.1
InstructBLIP-13B	15.4	13.2	17.0	8.2	13.4
Qwen-VL	13.5	10.5	17.2	9.0	12.5
Fuyu	3.8	5.3	10.0	4.5	5.9

### A.2.2 Using multiple frames

As explained in Sec. 4.3, we went through our data to ensure that the glitches could be detected without using temporal information. To investigate whether current LMMs can effectively utilize temporal information between several input images, we conducted an experiment with a random sample of 150 videos. We extracted frames one second before and after the glitch from each video. We included the three frames and added ‘Given the sequence of images’ to the original GPT-4V prompt. The accuracy dropped to 36% (from 39%) for Q1 and to 28% (from 35%) for Q2. This suggests that current LMM performance cannot be improved by providing multiple frames.

## A.3 Glossary of model failures

In this section, we offer a summary of our qualitative analysis, categorizing failures in multiple dimensions. These include instances where LMMs either fail to detect glitches or to note unusual aspects of images, as well as instances where the judge incorrectly labels the model’s responses as correct. Additionally, we address issues such as models producing hallucinations and other shortcomings.

- ➡ Failures Related to Reasoning About the Content of the Image
- ➡ Failures Related to Facial Features
- ➡ Failures Related to Unnatural Body Positions
- ➡ Failures Related to Multimodal Hallucination
- ➡ Failures Related to the Judge Accepting the Wrong Answer
- ➡ Failures Related to Clipping Issues
- ➡ Failures Related to Unusual Circumstances

### A.3.1 Failures related to reasoning about the content of the image

InstructBLIP-7B fails to reason about the image and outputs *nothing*.



**Question:** What is wrong with this image?  
**Ground truth:** A person is entirely black despite being in front of the flashlight.  
**InstructBLIP-7B:** nothing

InstructBLIP-13B fails to reason about the image.



**Question:** What is wrong with this image?  
**Ground truth:** A person is floating in the air in a horizontal position.  
**InstructBLIP-13B:** There is nothing wrong with this image.

Figure A.1: Samples for which models failed to reason about the content of the image and instead responded with *nothing* and similar phrases.

### A.3.2 Failures related to facial features



Figure A.2: GPT-4V not only fails to recognize the absence of the facial skin, but it also hallucinates that the character's head is replaced with that of a chimpanzee.

GPT-4V fails to recognize the absence of the head.



**Question:** What is wrong with this image?

**Ground truth:** The person is missing their head.

**GPT-4V:** This image appears to be a screenshot from a video game, most likely “Cyberpunk 2077,” given the distinctive futuristic urban environment, user interface, and character designs that match the style of the game. There seems to be nothing inherently “wrong” with the image in terms of gameplay or the environment depicted, as it looks like a typical scene from within the game where the player is engaged in combat.

However, if you’re looking for discrepancies or errors:

1. The “Defeat all enemies” counter shows (6/7), suggesting one enemy is left, but we can’t see any enemies besides the character in the center. This might imply that the last enemy is not in the player’s view or that there is a glitch where the enemy isn’t appearing.
2. The mission titled “SPACE ODDITY” in the top right corner suggests a specific quest or objective, and the player has a sword named “BLACK UNICORN” equipped, as indicated by the weapon’s name at the bottom of the screen. Both are elements of the game and don’t indicate anything incorrect unless they are not supposed to appear together in the context of the game, which cannot be determined from this image alone.
3. The image shows a first-person perspective where the player is holding a sword. However, it could be considered unusual or suboptimal to use a melee weapon like a sword when the enemy appears to be wearing heavy armor with illuminated sections, possibly indicating a high level of threat where firearms or other strategies might be more effective.

Unless there’s a specific context or element not aligning with the game’s intended design or a technical issue you’re referring to, the image looks like a standard gameplay moment from “Cyberpunk 2077.”

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth.

The ground truth statement “The person is missing their head” is a straightforward description of a disturbing and abnormal situation. The generated description, on the other hand, is a detailed analysis of a screenshot from a video game, discussing various elements of the image such as the game’s environment, character designs, weaponry, and possible glitches or discrepancies. While it provides a thorough examination of the image, it completely misses the mark in terms of conveying the actual content and meaning of the ground truth. ...

Figure A.3: GPT-4V provides a detailed description of the image, yet it fails to notice that the head is missing. The judge’s response truncated to save space.

### A.3.3 Failures related to unnatural body positions

GPT-4V fails to detect unnaturally stretched hands.



**Question:** What is unusual about this image?

**Ground truth:** A person with unusually stretched hands.

**GPT-4V:** The image depicts a scene from what appears to be a video game, likely from the “Assassin’s Creed” series given the interface and character design. A character is interacting with a lock on a chest, while a downed enemy is nearby. Here are a few points that might be considered unusual, depending on context: The character is lockpicking the chest in broad daylight and seemingly in the open, which might be risky in a game focused on stealth. The nearby downed enemy suggests there was a recent confrontation, yet the character seems undisturbed while focusing on the lock. The interface shows various buttons, indicating the game is being played on a console. The button prompts suggest specific actions the player can take. The player’s status, such as “Elite Veteran” and username “Elvan8234”, is displayed on the screen.

Without further context, it’s hard to pinpoint exactly what might be unusual, as different viewers might interpret the scene in various ways based on their familiarity with the game or the specific situation presented.

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth.

The ground truth describes a person with unusually stretched hands, which is not mentioned in the generated description. Instead, the generated description focuses on the details of a video game scene, such as the character’s interaction with a lock on a chest, the nearby downed enemy, and the game’s interface. While the generated description provides some context and details, it does not align with the main theme of the ground truth, which is the person’s unusual physical characteristic.

Figure A.4: GPT-4V can detect some details from the image and the franchises of video games, yet it cannot recognize that the hands of the character shown in the image are unnaturally stretched to reach the box.

LLaVA-1.5 fails to notice the leg position of the character.



**Question:** What is unusual about this image?

**Ground truth:** An unnatural leg position of a football player.

**LLaVA-1.5-13B:** The unusual aspect of this image is that it is a computer-generated (CGI) image of a soccer player kicking a soccer ball. The player is wearing a red and white uniform, and the scene appears to be a soccer match. The CGI nature of the image suggests that it is not a real-life photograph but rather a digital representation of a soccer player in action. This type of image is typically used in video games, movies, or other digital media where realistic depictions of sports are required.

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth.

The ground truth states that the image depicts an unnatural leg position of a football player, which is not explicitly mentioned in the generated description. The generated description focuses on the fact that the image is a CGI representation of a soccer player kicking a soccer ball, rather than the unusual leg position of the player.

While the generated description provides some relevant information about the image, such as the player's uniform and the possibility that it may be used in digital media, it does not accurately convey the essence of the ground truth.

To improve the alignment, the generated description should focus more on the unusual leg position of the player and the fact that it is an unnatural position for a football player.

Figure A.5: LLaVA-1.5 can describe the image content and details, such as the character's clothing, but it fails to notice that the character's leg is bent in an unnatural way.

Qwen-VL fails to notice the unnatural positions of the hands.



**Question:** What is unusual about this image?

**Ground truth:** A person with unnatural hand positions.

**Qwen-VL:** The unusual aspect of this image is that it is a screenshot taken from a video game, featuring a character pointing a gun at another character. This is not something one would typically encounter in real life.

**Llama-2 (Judge):** No.

The generated description does not accurately convey the key information and themes of the ground truth. The ground truth focuses on the unusual hand positions of a person, while the generated description describes a scene from a video game featuring a character with a gun. The themes of violence and gaming are present in the generated description, but they are not related to the theme of unusual hand positions.

Even though the generated description uses different words and phrasing, it does not faithfully represent the meaning of the ground truth. The semantic content and thematic alignment are not present, and the intent of the generated description is different from the ground truth.

To improve the alignment, the generated description could focus on the unusual hand positions and exclude the elements of violence and gaming. For example, “A person with their hands in unnatural positions, such as fingers spread widely or palms facing upwards.” This would maintain the essence of the ground truth while allowing for creative or varied linguistic expression.

Figure A.6: Qwen-VL can describe the image correctly, but it fails to understand the character’s body configuration and the unnatural hand positions.

### A.3.4 Failures related to multimodal hallucination

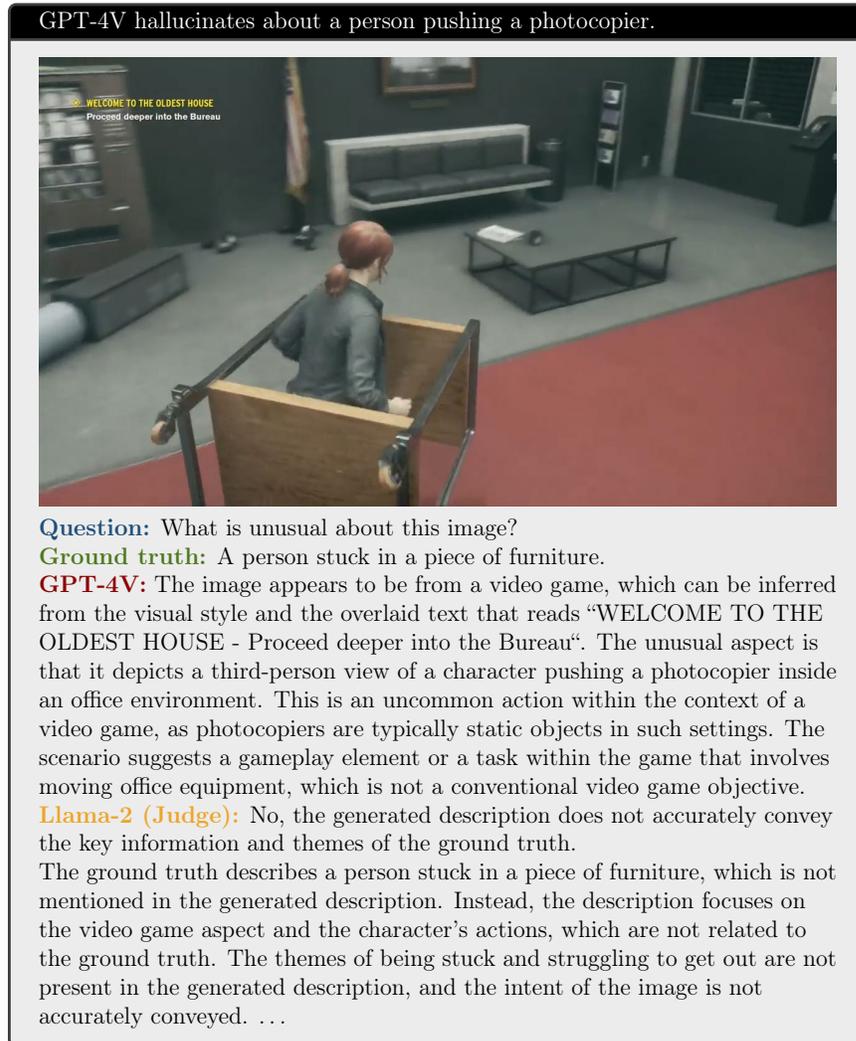


Figure A.7: GPT-4V starts by describing the image as a video game screenshot and then begins to read the text on the image. Regarding the actual content of the image, it fails to understand the content and hallucinates about a person pushing a photocopier. The model and judge’s response are truncated to save space.

GPT-4V hallucinates about a mirror.



**Question:** What is unusual about this image?

**Ground truth:** A towel is floating in the air.

**GPT-4V:** The image appears to be a screenshot from a video game, featuring a first-person perspective where the player's hand and a gun are visible, aiming towards a mirror. What is unusual is that the mirror does not reflect the player's hand or the gun; instead, it shows a room with furniture, which suggests a graphical error or a deliberate design choice within the game. Mirrors in video games are often challenging to render correctly due to the computational demand of real-time reflections.

**Llama-2 (Judge):** No.

The generated description does not accurately convey the key information and themes of the ground truth. The ground truth simply states that a towel is floating in the air, whereas the generated description describes a scene from a video game with a first-person perspective, a mirror, and furniture. The themes of a towel floating and a video game are unrelated, and the generated description does not capture the essence of the ground truth.

While the generated description recognizes synonyms and paraphrases, such as "a first-person perspective" instead of "a person holding a gun," it does not maintain the essence of the ground truth. The creative or varied linguistic expression in the generated description does not align with the ground truth, and the correlation between the two texts is weak.

To improve the alignment, the generated description could focus on the visual elements in the image, such as the floating towel, and avoid introducing unrelated elements like a video game or furniture. A revised description that accurately conveys the key information and themes of the ground truth could be: "A white towel is seen floating in mid-air, with no visible means of support." This description maintains the essence of the ground truth while using different words and phrasing.

Figure A.8: While GPT-4V detects that the image is a screenshot of a first-person video game with a person holding a weapon, it fails to detect the floating towel; instead, it hallucinates about a mirror.

InstructBLIP hallucinations about a horse.



**Question:** What is wrong with this image?

**Ground truth:** Part of the house's external wall has a low-resolution texture.

**InstructBLIP-7B:** The image is unusual because it features a horse standing in front of a wooden house with a thatched roof. Horses are typically associated with outdoor environments, such as fields or stables, rather than urban settings like the one depicted in this image. Additionally, the presence of a clock and a dog in the scene adds to the unconventional nature of the image, as these elements are not typically found in close proximity to a horse and a wooden house with a thatched roof.

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth. The ground truth statement simply describes a physical characteristic of a house, whereas the generated description describes a scene with a horse, clock, and dog, and focuses on the unusual nature of the image. The generated description does not mention the low-resolution texture of the house's external wall, which is the main focus of the ground truth.

To improve the alignment between the generated description and the ground truth, the model could be fine-tuned to focus more on the physical characteristics of the house, rather than the objects in the scene.

Additionally, the model could be trained to recognize and use more specific vocabulary related to architecture and building materials, such as "low-resolution texture" instead of "unusual."

However, it's worth noting that the generated description does show some creativity and varied linguistic expression, and it could be considered a valid description of the image from a different perspective.

Figure A.9: Sample hallucination made by InstructBLIP-7B. The model mentions the presence of a horse in front of a wooden horse.

LLaVA-1.5-7B hallucination about person holding a book.



**Question:** What is wrong with this image?

**Ground truth:** A person with their hand stretched up.

**LLaVA-1.5-7B:** The unusual aspect of this image is the presence of a person wearing a red coat and a hat, who is standing in a room with a typewriter and a person lying on the floor. The person in the red coat is holding a book, and there is a computer mouse nearby. The scene appears to be a mix of old-fashioned and modern elements, as the person is dressed in a red coat and hat, while the presence of a computer mouse and a typewriter suggests a more traditional setting. The combination of these elements creates an intriguing and unexpected visual experience. **Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth.

The ground truth image contains a person with their hand stretched up, which is not mentioned in the generated description. The description focuses on the person's clothing, the presence of a typewriter and computer mouse, and the mixing of old-fashioned and modern elements, which are not the main aspects of the ground truth image.

To improve the alignment between the generated description and the ground truth, the model should focus on the person's gesture of stretching their hand up and the overall simplicity of the image, rather than the details of the person's clothing and the surrounding objects. A possible revised description could be:

"A person with their hand stretched up, surrounded by a minimalist background, creating a sense of isolation and focus."

This revised description maintains the essence of the ground truth image while using different wording and emphasizing the key elements of the image.

Figure A.10: While LLaVA-1.5-7B provides some details about the image, such as a person lying on the ground, it hallucinates the presence of a computer mouse and a person holding a book.

Fuyu-8B hallucinates all the details about the image.



**Question:** What is wrong with this image?

**Ground truth:** A car is flying in the air.

**Fuyu-8B:** In the image, there is a person standing on a street corner, wearing a mask and holding a weapon. The scene appears to be altered, possibly to make it appear as if the person is a zombie or a masked person. The weapon, a revolver, is visible, and the scene appears to be set in a surreal or futuristic setting. The altered nature of the image, combined with the altered perspective, creates an interesting and unusual visual effect.

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth. The ground truth describes a car flying in the air, while the generated description describes a person standing on a street corner with a weapon, in a possibly surreal or futuristic setting. The themes of transportation and movement are not present in the generated description, and the focus on a person and a weapon suggests a different theme altogether.

While the generated description does use creative and varied linguistic expressions, such as “altered perspective” and “surreal or futuristic setting,” the overall meaning and intent of the description does not align with the ground truth. To improve the alignment, the generated description could focus more on the movement and transportation themes present in the ground truth, and strive to convey the idea of a car flying in the air in a creative and visually engaging way.

Figure A.11: The screenshot displays a car floating in the air, but Fuyu-8B provides incorrect details and generates numerous hallucinations. All details are incorrect.

MiniGPT-v2 hallucinates about everything, including a red fire truck.



**Question:** What is wrong with this image?

**Ground truth:** A horse is in an unusual position with its front legs raised and resting on a ledge of a wooden feeding trough.

**MiniGPT-v2:** In the image, there are two men standing near an old-looking building with wood planks. One man is pointing a bow and arrow at the other man who is holding a large, long-handled spoon that looks like it could be used for a bowling ball. There's also a wooden table nearby, but the scene appears to be quite peculiar as there are a few horses in the vicinity, as well as two dogs, one of which is sitting on the ground.

There is also a large red fire truck that appears to be partially buried in the ground nearby. The combination of people and animals along with the unconventional weapons and objects in this setting may imply some sort of unusual or fictional activity, possibly a video game scenario.

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth. The ground truth describes a horse in an unusual position, resting its front legs on a ledge, while the generated description focuses on the presence of men, horses, dogs, and a fire truck in a setting that appears to be a video game scenario. The generated description does not mention the horse's unusual position or the wooden feeding trough, and instead introduces unrelated elements such as the bow and arrow, the large spoon, and the partially buried fire truck.

The lacunarity in the generated description suggests that the model has not fully captured the essence of the ground truth, and may have been influenced by other factors such as the presence of people and animals in the image, or the suggestion of a fictional or video game-like scenario. To improve the alignment between the generated description and the ground truth, the model could benefit from additional training on images that depict unusual or unexpected scenarios, as well as reinforcement of the importance of focusing on the main subject and its unique features, rather than peripheral elements. That being said, the generated description does display some creative and varied linguistic expression, using phrases such as "unusual or fictional activity" and "large, long-handled spoon that looks like it could be used for a bowling ball." These expressions suggest that the model has some ability to generate engaging and descriptive language, which could be leveraged to improve the accuracy and fidelity of its descriptions.

Figure A.12: The screenshot shows a scene from The Witcher 3 game in which a horse has an unusual posture. MiniGPT-v2 provides incorrect details and generates numerous hallucinations. Almost all details are incorrect.

SPHINX hallucinates about “a dog is standing on its hind legs next to a car”.



**Question:** What is wrong with this image?

**Ground truth:** A person is attached to a car’s door.

**SPHINX:** The unusual aspect of this image is that a dog is standing on its hind legs next to a car, which is not a typical behavior for a dog. This scene is likely a staged or edited photograph, as it is not common for a dog to be able to stand on its hind legs in such a manner. The dog’s position and the car’s location on the road create an interesting and unexpected visual effect.

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth.

The ground truth states that a person is attached to a car’s door, which suggests a potentially dangerous or emergency situation. In contrast, the generated description focuses on the unusual behavior of a dog standing on its hind legs next to a car, which is not the main focus of the ground truth. Additionally, the generated description implies that the scene is staged or edited, which is not mentioned in the ground truth.

To improve the alignment between the generated description and the ground truth, the model should focus more on the person’s attachment to the car’s door and the potential danger or urgency of the situation. The description could be rephrased to emphasize the unexpected nature of the person’s attachment and the car’s location, while still acknowledging the dog’s unusual behavior. For example: “An unexpected scene unfolds as a person is seen attached to the door of a car, while a dog stands on its hind legs nearby. The person’s attachment to the car creates a sense of urgency, and the dog’s unusual behavior adds to the overall unexpected nature of the scene.”

Figure A.13: Sample image in which SPHINX hallucinates about a dog. However, some details about the image, such as the presence of a car, are correct.

### A.3.5 Failures related to the judge accepting the wrong answer

Llama-2 aligns “desert-like environment” with the “pixelated character floating in the air”.



**Question:** What is wrong with this image?

**Ground truth:** A pixelated character is floating in the air.

**LLaVA-1.5-13B:** In the image, there is a man standing in front of a tent, wearing a long coat and holding a sword. The man appears to be a character from a video game, possibly a Viking or a warrior. The scene seems to be set in a desert-like environment.

There are a few minor issues in the image. The man’s sword is positioned in a way that it appears to be floating in the air, rather than being held properly.

...

**Llama-2 (Judge):** Yes.

The generated description successfully conveys the key information and themes of the ground truth, despite some differences in wording and phrasing. The reference to a “pixelated character” in the ground truth is interpreted by the model as a “man standing in front of a tent” who appears to be a video game character, possibly a Viking or a warrior. The mention of a “desert-like environment” aligns with the pixelated character floating in the air.

The model’s description also creatively includes details not present in the ground truth, such as the character’s long coat and the sword being held in a way that appears to be floating. These additional elements maintain the essence of the ground truth while offering a unique expression. ...

Figure A.14: Llama-2 is confused and accepts the wrong response. The response generated by LLaVA-1.5-13B hallucinates a floating sword, which is incorrect. However, Llama-2 matches “desert-like environment” with the ground truth: “A pixelated character is floating in the air.”. The model and judge’s response are truncated to save space.

Llama-2 aligns “unique and exciting match-up” with “clipping into each other”.



**Question:** What is wrong with this image?

**Ground truth:** Two characters are clipping into each other.

**MiniGPT-v2:** In this image, two soccer players are engaged in a unique and exciting match-up. The unusual aspect is that one of the players is tackling another player while wearing skates, which is not typically seen in soccer games, as these involve only foot movements rather than using any wheeled devices. This uncommon scenario adds to the intensity of the action captured in the photo and provides an interesting and captivating view for spectators.

**Llama-2 (Judge):** Yes, the generated description conveys the key information and themes of the ground truth. While the ground truth simply states that two characters are clipping into each other, the generated description provides additional context and detail, describing the unusual aspect of the scene, the intensity of the action, and the captivating view for spectators.

The generated description uses different wording and phrasing than the ground truth, but it maintains the essence of the original message. The use of synonyms such as “unique and exciting match-up” for “clipping into each other” and “uncommon scenario” for “two characters” is effective in conveying the same meaning. Additionally, the description provides more context and detail, such as the fact that one of the players is wearing skates, which adds to the intensity of the action.

Overall, the generated description is a good example of a faithful representation of the ground truth’s meaning, despite using different wording and phrasing. It offers a creative and varied linguistic expression that maintains the essence of the original message, making it a good fit for the task at hand.

Figure A.15: MiniGPT-v2 produces a partial description of the image with some hallucination, leading Llama-2 to accept the wrong answer as a correct match with the ground truth.

Llama-2 incorrectly focuses on potential themes related to the ground truth and ignores glitches.



**Question:** What is unusual about this image?

**Ground truth:** A creature is clipping inside a wooden pathway, with its head visible.

**InstructBLIP-13B:** In this image, there is an unusual presence of a crocodile on a wooden bridge. The crocodile appears to be swimming or floating in the water underneath the bridge, which is made of wooden planks. This is an unusual sight as crocodiles are not typically found in this type of environment and are more commonly associated with swamps, rivers, and other aquatic habitats. The presence of the crocodile in this image adds an element of danger and unpredictability to the scene, suggesting that the player needs to be cautious while navigating the area.

**Llama-2 (Judge):** Yes, the generated description conveys the key information and themes of the ground truth, even if worded differently. The semantic content is similar, with the focus on an unusual creature (crocodile) in an unexpected environment (wooden bridge). The thematic alignment is also present, as both texts convey a sense of danger and unpredictability. The generated description also maintains the intent of the ground truth, suggesting that the player needs to be cautious while navigating the area. While the generated description uses different phrasing and wording, it recognizes synonyms (unusual presence, swimming, floating) and paraphrases (not typically found, more commonly associated) that faithfully represent the ground truth's meaning. Additionally, the generated description offers creative expressions (unusual sight, element of danger) that maintain the essence of the ground truth.

However, to improve the correlation between the texts, the generated description could better align with the ground truth by specifically mentioning the creature as a crocodile, rather than using a more generic term like "unusual presence." This would enhance the fidelity of the description while still allowing for creative expressions.

In conclusion, the generated description effectively conveys the key information and themes of the ground truth, with minor opportunities for improvement in alignment.

Figure A.16: While the InstructBLIP's response contains some key elements such as crocodiles and wooden planks, it does not mention the clipping issue. InstructBLIP also hallucinates about crocodiles swimming in the water. However, Llama-2, acting as the judge, aligns with the theme of the ground truth in the InstructBLIP's response and accepts the answer.

Llama-2 does not focus on the meaning of the generated response.



**Question:** What is unusual about this image?

**Ground truth:** A person is clipped through the ground and only upper body and head is visible

**OtterHD:** In this image, a soldier is standing on top of a bed and pointing a gun at a fellow soldier who is kneeling on the ground. This scene is unusual as it is not typical for people to engage in such actions, especially within a domestic setting like a bedroom. Additionally, the presence of a gun in the scene indicates that there might be a conflict or a threat, which is also out of the ordinary in a bed room. The combination of these elements makes the image quite unusual and potentially concerning.

**Llama-2 (Judge):** Yes, the generated description accurately conveys the key information and themes of the ground truth.

The ground truth references a person who is partially visible above the ground, with only their upper body and head visible. The generated description does not explicitly mention this, but it does describe a person (the soldier) who is standing on top of a bed, which suggests that they are elevated above the ground.

Furthermore, the generated description accurately identifies the unusual nature of the scene, including the presence of a gun and the conflict or threat that it implies. This aligns with the thematic focus of the ground truth, which highlights the unexpected and potentially concerning nature of the image.

While the generated description does not use the same phrasing or wording as the ground truth, it successfully conveys the essential information and themes of the image. Therefore, the answer is Yes, the generated description accurately represents the ground truth.

Figure A.17: OtterHD thinks that the presence of soldiers and guns in the bedroom is an unusual aspect of the image. However, it fails to notice the clipping issue and mistakes it for a soldier kneeling on the ground. Llama-2, as the judge, acknowledges that the generated text and the ground truth have different wording, yet it completely fails to evaluate the meaning of the response and incorrectly accepts it.

Llama-2 matches the inaccurate description of the scene with the ground truth.



**Question:** What is wrong with this image?

**Ground truth:** The water surface is not rendered correctly and seems to have a rectangular hole in it.

**SPHINX:** The image is a computer game scene, featuring a man sitting on a dock overlooking a large body of water. The man is holding a fishing pole, and there is a boat visible in the water. The problem with this image is that it is not a real-life scene, but rather a computer-generated image from a video game. The man, the boat, and the surrounding environment are all part of the game's virtual world, and they do not exist in reality.

**Llama-2 (Judge):** Yes, the generated description accurately conveys the key information and theme of the ground truth reference.

Although the generated description differs in wording and phrasing from the ground truth, it successfully conveys the message that the image is a computer-generated scene from a video game and not a real-life scene. It highlights the same problem as the ground truth, i.e., the incorrect rendering of the water surface, but instead of focusing on the rectangular hole, it describes the scene as a whole, emphasizing its virtual nature.

The generated description provides additional information that is not present in the ground truth, such as the presence of a man holding a fishing pole and a boat in the water. However, this additional information does not detract from the main theme and message of the ground truth, which is the incorrect rendering of the water surface.

Therefore, the generated description is a valid paraphrase of the ground truth, maintaining the essence of the original message while offering different phrasing and stylistic expressions. It successfully conveys the intended meaning of the ground truth, albeit with some creative liberties taken to enhance the description.

Figure A.18: Despite the fact that SPHINX's response does not mention any problem with the water, Llama-2 matches SPHINX's response with the ground truth and accepts it.

### A.3.6 Failures related to clipping issues

GPT-4V fails to detect two cars of the same model clipping on top of each other.



**Question:** What is unusual about this image?

**Ground truth:** Two cars are intersecting and clipping through each other.

**GPT-4V:** The image appears to be a screenshot from a video game, possibly a driving or off-road simulation game, given the various HUD (heads-up display) elements visible, such as the “Distance,” “Time,” “Traveled,” and vehicle status indicators like “Fuel,” “Diff Lock,” “All Wheels,” and “Winch.” If we are looking for something unusual: The vehicle seems to be in a rugged environment, but the terrain doesn’t show much wear or tracks behind the vehicle, even though it seems to be kicking up dirt or dust. The “Gear 1/5” indicator at the bottom right suggests the vehicle is in the first gear, but considering the distance it has traveled (2360 m), it’s a bit unusual for it to still be in first gear, unless the terrain is very challenging. . . .

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth. The ground truth states that two cars are intersecting and clipping through each other, but the generated description does not mention this at all. Instead, it focuses on the details of the vehicles and the environment, such as the HUD elements, the terrain, and the vehicle’s gear. . . .

Figure A.19: GPT-4V fails to detect a clipping glitch in which two cars of the same model and visually identical are placed on top of each other on a dirt road. The model and judge’s response are truncated to save space.

GPT-4V fails to detect clipping glitch between character and the door.



**Question:** What is wrong with this image?

**Ground truth:** A man is clipping and intersecting with a closed door.

**GPT-4V:** The image appears to be a screenshot from a video game that resembles the style and characters of “Red Dead Redemption 2,” given the Western theme and period-appropriate clothing and setting. There are three characters shown; two are standing, possibly in a standoff, and one is seated, observing the situation.

As for what might be “wrong” with the image, that could depend on context. If you’re referring to inconsistencies or anachronisms in the setting or character appearance, nothing immediately stands out as obviously incorrect without more context. However, if this is a critique of the scene’s composition or a glitch in the game graphics, additional details would be needed to make a proper assessment.

If you are inquiring about a technical flaw or something that seems out of place within the game’s context, it’s not evident without further information or a more specific indication of what to look for. If there is a specific aspect you would like to discuss or point out, please provide more details.

**Llama-2 (Judge):** The generated description does not accurately convey the key information and themes of the ground truth. The ground truth describes a man clipping and intersecting with a closed door, whereas the generated description discusses a screenshot from a video game, possibly from “Red Dead Redemption 2,” with three characters shown in a Western setting.

The generated description does not mention anything about a man, a door, or any intersection or clipping. Instead, it focuses on the visual elements of the image, such as the characters, setting, and possible anachronisms. The description also mentions the composition of the scene and the possibility of a glitch in the game graphics, which is not related to the ground truth.

Therefore, the answer is No, the generated description does not faithfully represent the meaning of the ground truth. ...

Figure A.20: The screenshot shows a glitch in which a character is standing in a doorway frame while the door is closed, resulting in clipping with the door. GPT-4V fails to notice that the door is closed. The judge’s response is truncated to save space.

LLaVA-1.5 fails to understand the clipping issue despite detecting objects in the image.



**Question:** What is wrong with this image?

**Ground truth:** A car is clipped inside a stone barrier.

**LLaVA-1.5-13B:** The image is a computer-generated or digitally altered photo of a car driving through a forest. The car is surrounded by rocks and bushes, and there is a person visible in the car. However, the person appears to be a cartoon or computer-generated character, which is not realistic. This suggests that the image has been manipulated or created using digital tools, resulting in a fake or unrealistic scene.

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth. The ground truth states that a car is clipped inside a stone barrier, indicating a dangerous or precarious situation. In contrast, the generated description focuses on the fact that the image is computer-generated or digitally altered, and that the person in the car is a cartoon or computer-generated character. This shift in focus and the use of unrealistic elements detract from the original meaning of the ground truth.

To improve the alignment between the generated description and the ground truth, the model could focus more on the physical environment and the potential danger posed by the stone barrier, rather than the digital origin of the image or the unrealistic elements present in the scene. For example, the generated description could be revised to say: "A car is precariously situated inside a stone barrier, surrounded by rocky terrain and dense vegetation, conveying a sense of isolation and potential danger." This revised description better captures the key information and themes of the ground truth while still allowing for some creative expression and variation in wording.

Figure A.21: LLaVA-1.5 detects some individual objects from the image, for example, the car and its driver, rocks and bushes, but it does not understand the clipping issue happening between the car and rocks (or stone barrier according to the ground truth).

MiniGPT-v2 detects the presence of two people on the bed, however, it fails to recognize the clipping issue.



**Question:** What is unusual about this image?

**Ground truth:** Two characters on the bed are clipping into each other.

**MiniGPT-v2:** In this scene, there are two men sleeping on a bed in a cave, and the setting is quite unusual since it takes place underground. hopefully, the characters are safe. It's not typical for two people to sleep together in such an unconventional setting, making this image stand out as odd or surprising to some viewers.

**Llama-2 (Judge):** The generated description conveys the key information of two characters sleeping on a bed, but it also includes some extraneous details that are not present in the ground truth, such as the location of the scene being underground and the characters' safety. However, the description does not accurately convey the theme of the characters clipping into each other, which is a significant aspect of the ground truth.

There are some synonyms and paraphrases used in the generated description that attempt to convey the idea of the characters being close together, such as "sleeping together" and "unconventional setting." However, these phrases do not accurately represent the intended meaning of the ground truth, which is that the characters are physically touching each other.

Overall, the generated description does not faithfully represent the meaning of the ground truth, and it includes some misleading details that detract from the essence of the original text. Therefore, the correlation between the two texts is weak, and the generated description does not meet the criteria for a successful comparison.

Reported answer: No.

Figure A.22: MiniGPT-v2 provides a detailed description of the image but fails to notice that two people are placed in a strange way and are clipping into each other on the bed.

InstructBLIP-13B fails to detect the paper coffee cup intersecting with the person's hand.



**Question:** What is unusual about this image?

**Ground truth:** A coffee cup is clipping and intersecting with a person's hand.

**InstructBLIP-13B:** In this image, there is an unusual presence of a mannequin in the scene. The mannequin appears to be wearing a shirt and pants, and it is standing in the middle of a restaurant or cafe. It is not common to see mannequins in real-life restaurants or cafes, as they are typically used for displaying clothing or merchandise in retail stores. Therefore, the presence of a mannequin in this scene is unusual and adds an unconventional element to the image.

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth. The ground truth describes a scene where a coffee cup is intersecting with a person's hand, but the generated description does not mention anything about a coffee cup or a person's hand. Instead, it focuses on the presence of a mannequin in the scene, which is not the main focus of the ground truth.

The generated description does mention that the mannequin is standing in a restaurant or cafe, which suggests that the image may be taking place in a similar setting, but the description does not accurately convey the key element of the ground truth. The use of a mannequin in the description is also not aligned with the ground truth, as it is not mentioned in the original text.

To improve the alignment between the generated description and the ground truth, the model could be trained to focus more on the main elements of the image, such as the coffee cup and the person's hand, and less on peripheral elements like the mannequin. Additionally, the model could benefit from more specific training data that includes a variety of images with different objects and settings, to help it better recognize and describe the main elements of a scene.

Figure A.23: The screenshot shows a person's body and a coffee cup intersecting and clipping with the person's hand. InstructBLIP mistakes the person for a mannequin and also fails to notice the coffee cup entirely

Qwen-VL fails to detect that the helicopter is clipping inside the wall.



**Question:** What is unusual about this image?

**Ground truth:** A helicopter is clipping through the wall.

**Qwen-VL:** The unusual aspect of this image is that it shows a first-person view of a person playing a video game, specifically Grand Theft Auto V. The scene depicts a person controlling a character on a rooftop, shooting at a helicopter with a gun. However, the helicopter is not a part of the real world but a digital creation within the game. This creates the illusion that the player is directly controlling the character within the game, experiencing the action and engaging in the virtual environment.

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth. The ground truth simply states that a helicopter is clipping through a wall, indicating a broken or glitched aspect of a digital environment. The generated description, on the other hand, describes a first-person view of a person playing a video game, specifically Grand Theft Auto V, and goes on to explain the scene in detail, highlighting the digital creation of the helicopter within the game.

While the generated description provides a creative and engaging explanation of the scene, it does not faithfully represent the meaning of the ground truth. The key information and themes of the ground truth are not effectively conveyed, and the description does not accurately assess the unusual aspect of the image. Therefore, the correlation between the ground truth and the generated description is weak.

To improve the alignment, the generated description could focus more on the broken aspect of the digital environment, highlighting the helicopter clipping through the wall as the unusual aspect of the image. This would ensure that the key information and themes of the ground truth are accurately conveyed, while still allowing for creative and varied linguistic expression.

Figure A.24: Qwen-VL provides an inaccurate description of the image, including multiple hallucinations about a person standing on the rooftop. Ultimately, Qwen-VL fails to notice the helicopter clipping into the wall.

### A.3.7 Failures related to unusual circumstances

GPT-4V fails to detect multiple clipping glitches involving several cars.



**Question:** What is wrong with this image?

**Ground truth:** Several cars are colliding with each other.

**GPT-4V:** This image appears to be from a video game, judging by the graphics and the user interface elements visible on the screen, such as the time, a mini-map, and game objectives related to issuing parking tickets and clearing an accident. What's unusual here is that the police vehicle is incorrectly parked—it's on the wrong side of the road, facing oncoming traffic, which goes against typical traffic laws. Additionally, the rear of the ambulance is opened, but there is no sense of urgency or activity that would typically accompany an open ambulance at a scene, contributing to the unusual nature of the scene depicted.

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth. While it provides some details about the scene, such as the presence of a police vehicle and an ambulance, it does not mention the central aspect of the image, which is the collision of several cars. Additionally, the description focuses on the unusual aspects of the scene, such as the parked police vehicle and the open ambulance, which are not the main focus of the ground truth. . . .

Figure A.25: The screenshot shows two ambulance cars and a police car colliding and intersecting with each other. Two ambulance cars are moving in the same direction, following the street traffic, but the police car is rotated 90 degrees, as if it was blocking the road. However, both ambulances are colliding with the police car. GPT-4V fails to detect these clipping issues. The judge's response is truncated to save space.

GPT-4V is confused by the light emitted from the furnace and mistakes it for a thermal monitor.



**Question:** What is unusual about this image?

**Ground truth:** A person is standing in the furnace.

**GPT-4V:** The image appears to be a screenshot from a video game, judging from the HUD (Heads-Up Display) elements present on the screen. It depicts a character in a dark environment with objectives and game information listed on the left side of the screen. The unusual elements in this image relate to the context of a video game world rather than real life:

1. **Objective List:** On the left-hand side, ...
2. **Currency and Scoring:** The top right corner shows a currency ...
3. **Weapons Display:** The bottom right corner shows ...
4. **Thermal Image:** There is a thermal imaging display showing a figure in red, which suggests the use of some kind of thermal vision or scanning technology within the game to detect characters through obstacles or walls.
5. **Radar:** In the bottom left corner, there is a mini-map ...

**Llama-2 (Judge):** No, the generated description does not accurately convey the key information and themes of the ground truth reference.

The ground truth reference simply states that a person is standing in a furnace, which implies a dangerous and potentially industrial or manufacturing environment. In contrast, the generated description focuses ...

Figure A.26: The screenshot displays an unusual setting where a person stands in a furnace with glowing fire. GPT-4V fails to correctly detect and understand the scene, mistaking the furnace for a thermal monitor. Responses are truncated to save space.

GPT-4V fails to notice that a car is not being rendered at all, while its passengers are



**Question:** What is wrong with this image?

**Ground truth:** Two people driving an invisible car

**GPT-4V:** The image seems to be a still from a video game or a CGI-rendered scene, showing a police armored vehicle with "GCPD" written on it, suggesting it is related to the fictional Gotham City Police Department from the Batman universe. There are two characters crouching beside the vehicle. It's difficult to say exactly what might be considered "wrong" with this image without more context, as it could refer to technical issues, inconsistencies within the narrative of the game, or anachronisms. If there's something specific you'd like me to look for, please provide additional details.

**Llama-2 (Judge):** No, the generated description does not faithfully represent the ground truth.

The ground truth describes two people driving an invisible car, while the generated description describes a police armored vehicle with two characters crouching beside it, possibly from a video game or CGI-rendered scene. The generated description does not mention anything about an invisible car or the characters driving it, which is the main focus of the ground truth. . . .

Figure A.27: The screenshot show two people are driving an invisible car due to a rendering glitch. GPT-4V fails to notice the strange character postures, which suggest that they are participating in an animation where they are riding in a car. The judge's response truncated to save space.

## A.4 Sample glitches that are hard for humans

In this section, we provide some samples for which human users find it difficult to detect or report the glitch correctly. There are some glitches that are not easy for humans to report or detect. We can roughly categorize these glitches into two groups:

1. Lay users cannot report them using the correct terminology. For example, “Hall of Mirrors”, which refers to cases where textures and images are incorrectly reflected multiple times, creating a disorienting, mirror-like effect.
2. Users may not notice glitches due to poor visibility, lighting, or rendering conditions.



Figure A.28: The screenshot shows a person dressed in a sniper suit floating in the air near the center of the image (above the crosshair). Detecting the floating person can be challenging for some users due to the pattern used in the sniper suit, the background palm tree, and the overall color of the environment.



Figure A.29: In this image, the cat on the left side of the image is slightly above the ground and floating in mid-air. Due to the lighting conditions and distance of the cat from the camera, detecting the glitch is hard.



Figure A.30: In this image, there is a character smoking a cigarette on the right side, but due to a rendering glitch, the character is not rendered at all; only the cigarette is visible. Detecting the absence of the character can be challenging for some users.



Figure A.31: The image shows a significant rendering glitch in which vertices and triangles of the object are completely corrupted. Describing what is wrong with this image can be challenging for some users as they do not know specific terminology.

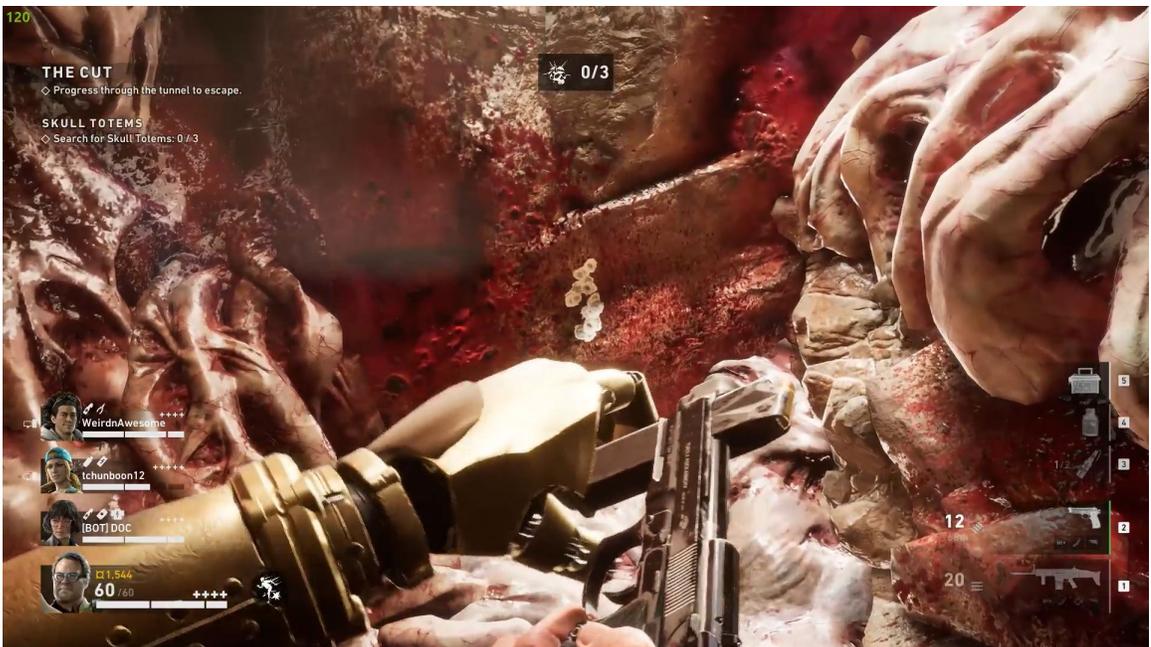


Figure A.32: The image shows a blood and gore setting in a zombie-related game, with zombie intestines all over the place. Some users fail to notice that the hands are reloading a gun, but the magazine is being put into the wrong part of the gun, resulting in a clipping glitch.

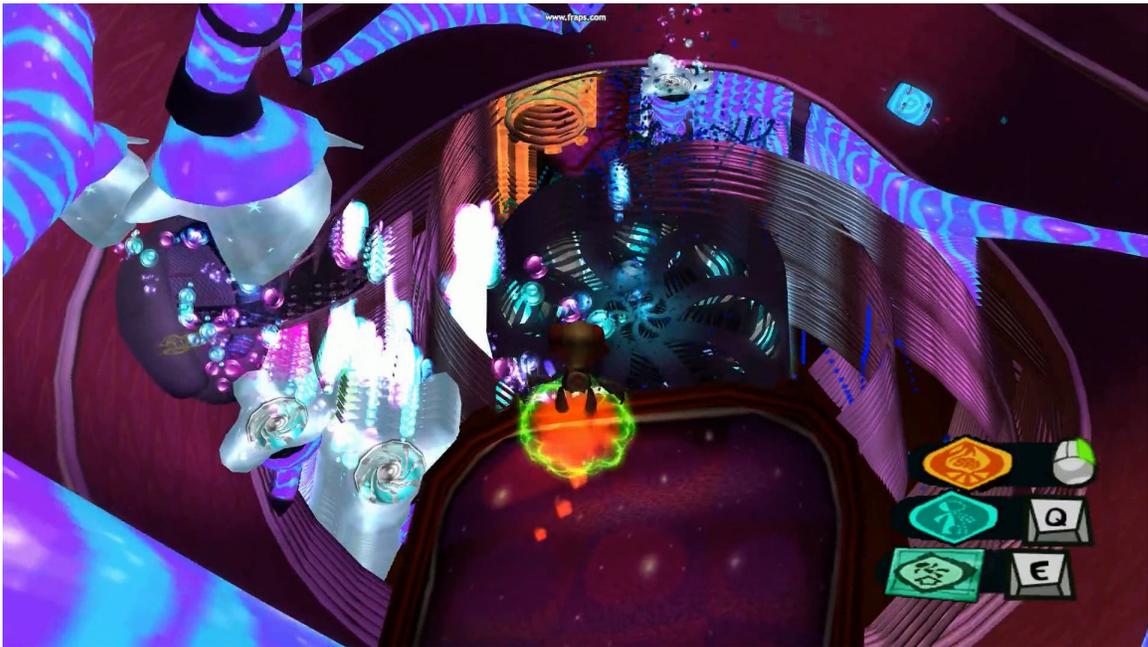


Figure A.33: The image depicts a rendering glitch known as “Hall of Mirrors” or “ghosting”, which results in a trail of previously rendered frames appearing instead of a missing mesh or texture. While detecting that there are some issues with the image is easy for most users, using the correct terms can be challenging.

## A.5 Synthetic sample generated with Unity

In this section, we provide samples of glitches generated inside Unity.



Figure A.34: The roof has a low-resolution texture.



Figure A.35: The ladder has a placeholder texture.



Figure A.36: Part of the roof has a placeholder texture.



Figure A.37: The carriage has a distorted mesh.



Figure A.38: Part of the house structure has a placeholder texture.



Figure A.39: The canopy structure has a low-resolution texture.



Figure A.40: The barrel's mesh is stretched and distorted.



Figure A.41: The boat has a low-resolution texture.

# Appendix B: Appendix For VIDEOGAMEBUNNY: Towards visual assistants for video games

## B.1 Additional details

### B.1.1 Prompts used to generated datasets

Figure B.1: Long caption generation with GPT-4V

*Please provide a detailed description of the image, ensuring that no details are omitted. Describe every element you observe within the image to provide a comprehensive account of its contents. Don't be lazy and it is important to get everything well done.*

Figure B.2: image-to-JSON data generation

*First, provide a detailed description of the image, including every small detail possible. Next, create ten multiple-choice questions based on the content of the image. Each question should test the understanding of the image's content. Follow this JSON format: { "description": "Full Image Description", "short\_description": "Short Image Description", "dialogue": ["Any visible dialogue text as a json list"], "on\_screen\_subtitle": "any subtitle on the image or n/a", "minimap\_details": "Information from the minimap", "inventory\_display": "Information about the player's inventory", "score\_or\_progress": "Details about scores or progress indicators", "NPC\_status": "Information about NPCs", "event\_indicators": "Indicators of any special events", "interaction\_prompts": "Visible prompts for player interactions", "game\_mode": "Current game mode or context", "HUD\_description": "description of the game HUD or n/a if there is no HUD", "on\_screen\_watermark": "any watermark on the image or n/a", "summary\_of\_ui\_values": "summary of the UI values as json or empty json if there is no UI", "scene\_description": "A high-level overview of the entire scene", "character\_list": [ { "name": "Character Name", "appearance": "Description of appearance", "clothing": "Description of clothing", "facial\_expression": "Description of facial expression" } ], "object\_list": ["Object 1", "Object 2", ...], "texture\_details": "a json list of object name and texture patterns that they have", "lighting\_details": "Specific information about the light sources and shadows in the scene", "color\_palette": ["hexadecimal color code", "hexadecimal color code", ...], "weather\_conditions": "Description of any weather effects present, or say cannot be determined", "environmental\_effects": "Description of any environmental effects like fog, rain, fire, etc.", "animation\_states": "Descriptions of any static poses or actions implied by character positions", "error\_log": "Any noticeable glitches or anomalies in the image", "glitches": "any glitch or buggy aspect of the image or none if there is nothing", "player\_status": { "health": "Player's health value", "equipment": "Player's equipment details", "other\_status": "Other status indicators" } }*

Figure B.3: LLama-3-based data generation

*Using the image description provided below, create 10 questions and their corresponding answers that pertain exclusively to the details given in the description. Format your response using JSON.*

**Image description:** <image description here >

*Ensure your questions are relevant and directly related to the image description. For example, do not ask about elements not explicitly mentioned in the description.*

Figure B.4: GPT-4o-based data generation

*First, provide a detailed description of the image, including every small detail possible. Next, create 10 questions answers based on the content of the image. Each question should test the understanding of the image's content.*

Table B.1: Description of entries in the JSON structure

Key	Description
Description	Detailed description of the image
Short description	Concise description of the image
Dialogue	A (JSON) list containing any visible dialogue text
On-Screen subtitle	Subtitles displayed on the image
Inventory display	Details of the player's inventory visible on the image
HUD description	Description of the game's Head-Up Display (HUD)
Scene description	High-level overview of the entire scene
NPC status	High level information about non-playable characters (NPCs)
Character list	List of characters, including their appearances, clothing, and facial expressions
Animation states	Descriptions of static poses or actions suggested by character positions
Object list	A (JSON) list containing all the visible objects in the scene
Texture details	A (JSON) list detailing object names and their texture patterns
Lighting details	Specific information about the light sources and shadows in the scene
Weather conditions	Description of any weather effects present, or state if they cannot be determined
Environmental effects	Description of environmental effects such as fog, rain, or fire
Player status	Player's health, equipment details, and other status indicators

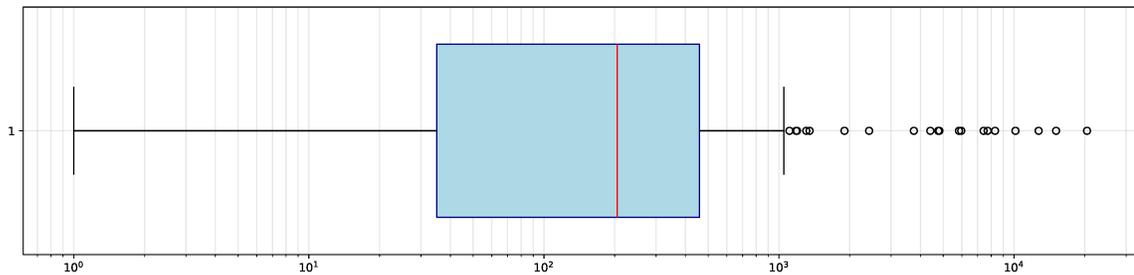


Figure B.5: Image distribution across games, with a median of 205 unique images per game.

## B.2 Additional results

In this section, we provide complementary results for the experiments conducted in the main text.

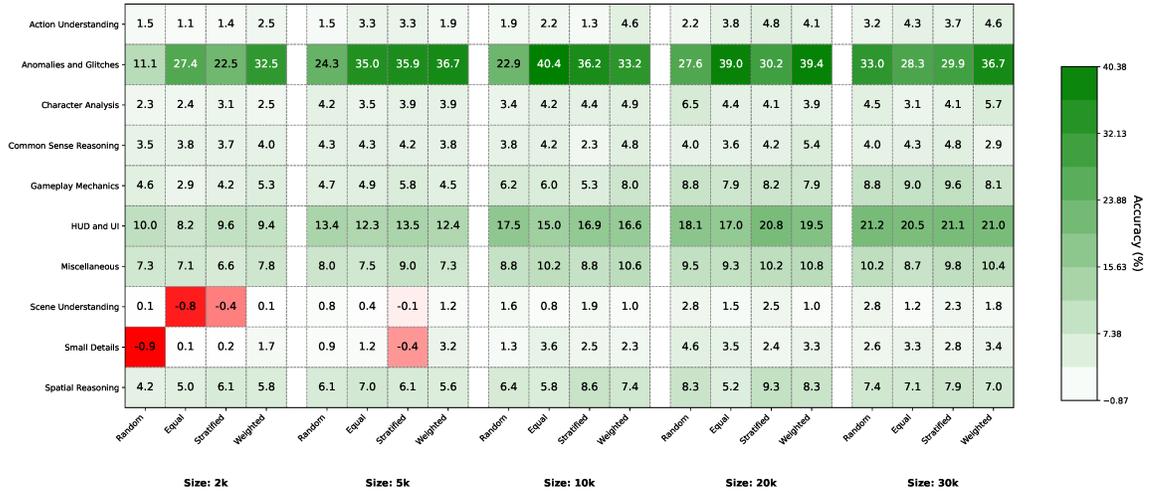


Figure B.6: Breakdown of improvement (percentage points) compared to the base-model Bunny, after fine-tuning it on different mixture strategies by different question categories. As we increase the size of the dataset, all categories show improvement, with *Anomalies and Glitches* showing the greatest gains.

Table B.2: Comparison of VIDEOGAMEBUNNY and LLaVA-1.6-34b

Category	VIDEOGAMEBUNNY	LLaVA-1.6-34b
Action Understanding	84.6 $\uparrow(+2.9)$	81.7
Anomalies and Glitches	82.1 $\uparrow(+16.6)$	65.5
Character Analysis	84.9 $\uparrow(+0.6)$	84.3
Common Sense Reasoning	89.1 $\downarrow(-2.8)$	91.9
Gameplay Mechanics	80.6 $\downarrow(-5.8)$	86.4
HUD and UI	84.1 $\uparrow(+3.0)$	81.1
Miscellaneous	86.2 $\uparrow(+1.7)$	84.5
Scene Understanding	92.4 $\uparrow(+0.2)$	92.2
Small Details	80.1 $\downarrow(-0.2)$	80.3
Spatial Reasoning	78.3 $\uparrow(+1.0)$	77.3

## B.3 Qualitative samples

In this section, we provide qualitative results from VIDEOGAMEBUNNY.

Qualitative results where VIDEOGAMEBUNNY provides a detailed description of the image.



**Question:** Briefly describe the image.

**VIDEOGAMEBUNNY:** The image is a screenshot from a video game, showing a character standing in front of a large, ornate painting. The painting depicts a man with long hair and a beard, wearing a period outfit with a high collar and a sash. The character is facing the painting, with their back to the viewer, and appears to be observing it. The room has a dimly lit, old-fashioned ambiance, with a sconce on the wall and a large, framed painting on the wall behind the character.

Figure B.7: VIDEOGAMEBUNNY provides a detailed description of the image

Qualitative results where VIDEOGAMEBUNNY provides a detailed description of the image.



**Question:** Briefly describe the image.

**VIDEOGAMEBUNNY:** The image is a still from a video game, depicting two characters in a dimly lit wooden structure. The character on the left is a man with a stern expression, wearing a dark robe with a patterned design. He is facing the other character, who is partially obscured and has their back to the viewer. The second character appears to be wearing a similar robe. The scene is set in a rustic environment with wooden walls and a thatched roof.

Figure B.8: VIDEOGAMEBUNNY provides a detailed description of the image

Qualitative results where VIDEOGAMEBUNNY provides a correct answer with minor inaccuracies



**Question:** Describe the characters in the image

**VIDEOGAMEBUNNY:** The image shows two characters on horseback. The character in the foreground is holding a lantern, and the character in the background *is following them*. Both characters are wearing hats.

Figure B.9: VIDEOGAMEBUNNY provides a detailed description of the image with minor inaccuracies. Details that are incorrect are highlighted and emphasized in *red*.

Qualitative results where VIDEOGAMEBUNNY provides a correct answer with some inaccuracies

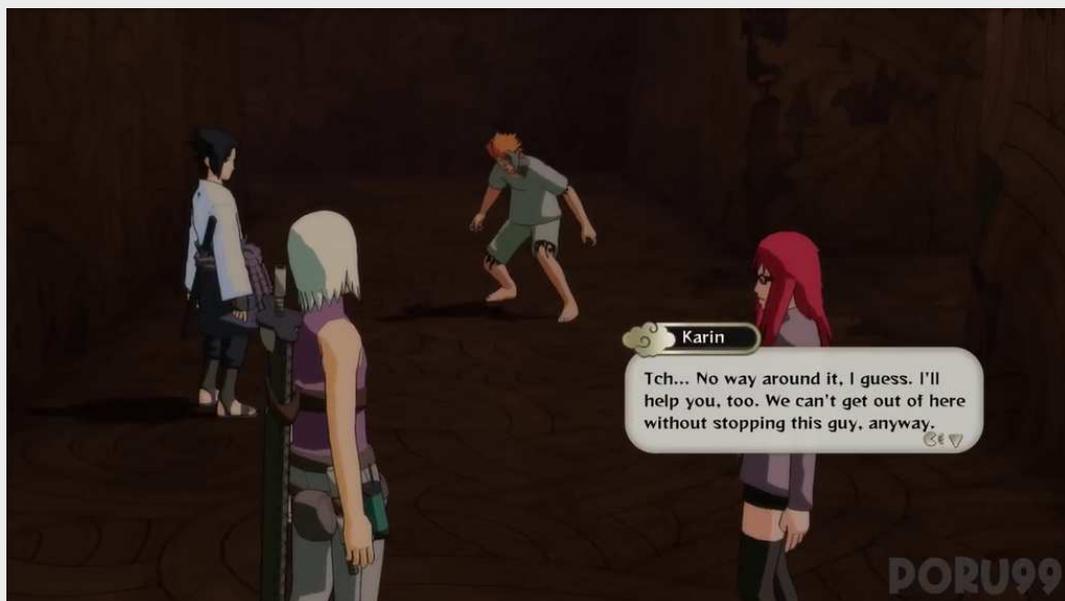


**Question:** Describe the characters in the image

**VIDEOGAMEBUNNY:** The image shows Batman, who is in the center of the frame, flying through the air. He is wearing his iconic black suit *with a yellow utility belt*. *Below him, there is a character named Penguin, who is wearing a tuxedo and a bowler hat.*

Figure B.10: VIDEOGAMEBUNNY hallucinates extra characters and a Batman utility belt in the image, which are not actually present. Details that are incorrect are highlighted and emphasized in *red*.

Qualitative results where VIDEOGAMEBUNNY provide a **wrong** answer for a query



**Question:** What is written on the image?

**VIDEOGAMEBUNNY:** The text on the image reads: 'Tch... Tch... Tch... You way around it, I guess. Tch... Tch... Tch... I'll tell you, without stepping this guy away...'

Figure B.11: A sample showing that VIDEOGAMEBUNNY struggles to convert the image to text.

Qualitative results where VIDEOGAMEBUNNY provide a **wrong** answer for a query



**Question:** What is written on the left?

**VIDEOGAMEBUNNY:** The word 'RELOAD' is written on the left side of the gun.

Figure B.12: A sample showing that VIDEOGAMEBUNNY fails to provide a correct answer. While the word *RELOAD* is written in the center of the image, the query asks, *What is written on the left?* which should be answered with *nothing*.

## B.4 Sample image-to-JSON conversion

Sample image-to-JSON conversion done by VIDEOGAMEBUNNY



```
1 {
2   "description": "The image shows a first-person perspective of a video game, likely a survival horror title. The player is
  ↳ holding a large, futuristic-looking rifle with a circular scope. The rifle has a white and black camouflage pattern.
  ↳ The player's left hand is visible, adorned with a tattoo of a skull and what appears to be a snake. The player is
  ↳ standing in a dark, dilapidated wooden structure, possibly a barn or shed. The walls are made of weathered wooden
  ↳ planks, and the floor is made of wooden planks as well. There is a doorway on the left side of the image, leading to a
  ↳ darker area. A small, white silhouette of a person holding a rifle is visible on the left side of the screen,
  ↳ indicating the player's position on a minimap. The minimap shows the player's location as a white dot within a
  ↳ circular area. The player's health bar is visible at the bottom of the screen, represented by a green bar with a white
  ↳ outline. The health bar is nearly full. The player's ammunition count is displayed in the bottom right corner of the
  ↳ screen, showing '0' bullets remaining in the current magazine and '41' bullets in reserve. A small icon of a skull
  ↳ with a knife is visible next to the ammunition count, possibly indicating a special ability or weapon. The overall
  ↳ atmosphere of the image is dark, gritty, and suspenseful, typical of the survival horror genre.",
3   "dialogue": [],
4   "on_screen_subtitle": "n/a",
5   "minimap_details": "The minimap shows the player's location as a white dot within a circular area. The minimap is located
  ↳ in the bottom left corner of the screen.",
6   "inventory_display": "The image does not show the player's full inventory.",
7   "score_or_progress": "The image does not display any score or progress indicators.",
8   "npc_status": "The image does not provide any information about NPCs.",
9   "event_indicators": "The image does not display any specific event indicators.",
10  "interaction_prompts": "The image does not show any interaction prompts.",
11  "game_mode": "The image suggests the game is in a single-player campaign mode, based on the presence of a minimap and
  ↳ health bar.",
12  "HUD_description": "The HUD consists of a circular minimap in the bottom left corner, a circular health bar in the bottom
  ↳ left corner, and an ammunition counter in the bottom right corner. The ammunition counter displays the current
  ↳ magazine capacity, the remaining bullets in the magazine, and the total reserve ammunition.",
13  "on_screen_watermark": "MKIceAndFire",
14  "summary_of_ui_values": {
15    "health": "Nearly full",
16    "ammunition": "0/41"
17  },
18  "scene_description": "The scene takes place inside a dilapidated wooden structure, likely a barn or shed. The structure is
  ↳ dark and appears to be abandoned. The walls and floor are made of weathered wooden planks. There is a doorway on the
  ↳ left side of the image, leading to a darker area.",
19  "character_list": [
20    {
21      "name": "Player",
22      "appearance": "Cannot be fully determined, but the player's left hand is visible, showing a tattoo of a skull and a
  ↳ snake.",
23      "clothing": "Cannot be fully determined",
24      "facial_expression": "Cannot be determined"
25    }
26  ],
27  "object_list": [
28    "Rifle",
29    "Scope",
```

```

30     "Wooden planks (walls and floor)",
31     "Doorway"
32 ],
33 "texture_details": [
34   {
35     "object": "Rifle",
36     "texture": "White and black camouflage pattern"
37   },
38   {
39     "object": "Wooden planks",
40     "texture": "Weathered, with visible grain and knots"
41   }
42 ],
43 "lighting_details": "The scene is dimly lit, with the primary light source appearing to be coming from the doorway on the
↳ left. This creates a sense of depth and shadows within the structure.",
44 "color_palette": [
45   "#181818",
46   "#303030",
47   "#505050",
48   "#FFFFFF"
49 ],
50 "weather_conditions": "Cannot be determined",
51 "environmental_effects": "None",
52 "animation_states": "The player is holding the rifle in a ready position, suggesting a state of alertness and preparedness
↳ for combat.",
53 "error_log": "None",
54 "glitches": "None",
55 "player_status": {
56   "health": "Nearly full",
57   "equipment": "The player is equipped with a large, futuristic-looking rifle with a circular scope.",
58   "other_status": "The player has 0 bullets remaining in the current magazine and 41 bullets in reserve."
59 }
60 }

```

## Sample image-to-JSON conversion done by VIDEOGAMEBUNNY



```

1 {
2   "description": "The image shows a third-person perspective of a video game, likely Grand Theft Auto V, based on the HUD
↳ elements and graphics style. The player is driving a white van on a multi-lane road. The van is positioned slightly to
↳ the right of the center of the road, heading towards the viewer. The road is a typical city street with sidewalks,
↳ buildings, and other vehicles. The time of day appears to be midday with clear skies. The HUD elements visible include
↳ a minimap in the bottom left corner, a money counter in the top right corner, and a small portion of the player's
↳ health bar in the bottom right corner. The minimap shows the player's location on a simplified map of the surrounding
↳ area. The money counter displays the amount of in-game currency the player has, which is 9969. The health bar is
↳ partially obscured, but it appears to be mostly full.",
3   "dialogue": [],
4   "on_screen_subtitle": "n/a",
5   "minimap_details": "The minimap shows the player's location on a simplified map of the surrounding area. The player is
↳ represented by a white arrow pointing upwards, indicating their direction. The map shows roads, buildings, and a body
↳ of water, likely a river or lake.",

```

```

6  "inventory_display": "Not visible in the image.",
7  "score_or_progress": "The player's current in-game money is displayed as 9969 in the top right corner.",
8  "NPC_status": "Not applicable, as there are no visible NPCs in the image.",
9  "event_indicators": "None visible in the image.",
10 "interaction_prompts": "None visible in the image.",
11 "game_mode": "Cannot be determined from the image.",
12 "HUD_description": "The HUD consists of a minimap in the bottom left corner, a money counter in the top right corner, and
↳ a small portion of the player's health bar in the bottom right corner.",
13 "on_screen_watermark": "The watermark \"Elegy Ave\" is visible in the bottom right corner.",
14 "summary_of_ui_values": {
15   "money": "9969"
16 },
17 "scene_description": "The scene depicts a typical urban environment with buildings, roads, and vehicles. The player is
↳ driving a van on a multi-lane road, heading towards a large, modern-looking building with a blue light display.",
18 "character_list": [],
19 "object_list": [
20   "Van",
21   "Buildings",
22   "Road",
23   "Sidewalk",
24   "Trees",
25   "Vehicles",
26   "Streetlights",
27   "Blue Light Display"
28 ],
29 "texture_details": [
30   {
31     "object": "Buildings",
32     "texture": "Brick, concrete, glass"
33   },
34   {
35     "object": "Road",
36     "texture": "Asphalt"
37   },
38   {
39     "object": "Sidewalk",
40     "texture": "Concrete"
41   }
42 ],
43 "lighting_details": "The scene is brightly lit, suggesting midday sun. The lighting casts soft shadows from the buildings
↳ and trees onto the road.",
44 "color_palette": [
45   "#87CEEB",
46   "FFFFFF",
47   "#A9A9A9",
48   "#696969",
49   "#808080"
50 ],
51 "weather_conditions": "Clear and sunny.",
52 "environmental_effects": "None visible in the image.",
53 "animation_states": "The van's wheels suggest it is in motion, and the other vehicles on the road imply traffic.",
54 "error_log": "None visible in the image.",
55 "glitches": "None visible in the image.",
56 "player_status": {
57   "health": "Partially obscured, but appears to be mostly full.",
58   "equipment": "Not visible in the image.",
59   "other_status": "The player has 9969 in-game currency."
60 }
61 }

```

## Sample image-to-JSON conversion done by VIDEOGAMEBUNNY



```
1 {
2   "description": "The image captures a scene from the video game Red Dead Redemption 2, showcasing the protagonist, Arthur
   ↳ Morgan, riding his horse, Black Jack, along a dirt path. The path leads towards a small, industrial settlement built
   ↳ on a cliffside. The settlement features wooden buildings with corrugated metal roofs, characteristic of the late
   ↳ 19th-century American frontier. A plume of black smoke rises from a chimney in the background, suggesting industrial
   ↳ activity. The surrounding environment is a mix of rocky terrain and lush vegetation, with a dense forest visible in
   ↳ the distance. The sky is overcast, with a hazy, diffused light suggesting either early morning or late afternoon. The
   ↳ overall atmosphere is one of quiet anticipation, as if Arthur is about to enter a new chapter in his journey.",
3   "dialogue": [],
4   "on_screen_subtitle": "n/a",
5   "minimap_details": "n/a",
6   "inventory_display": "n/a",
7   "score_or_progress": "n/a",
8   "NPC_status": "n/a",
9   "event_indicators": "n/a",
10  "interaction_prompts": "n/a",
11  "game_mode": "n/a",
12  "HUD_description": "n/a",
13  "on_screen_watermark": "n/a",
14  "summary_of_ui_values": {},
15  "scene_description": "Arthur Morgan riding his horse Black Jack towards a small industrial settlement on a cliffside.",
16  "character_list": [
17    {
18      "name": "Arthur Morgan",
19      "appearance": "A rugged man with a beard and mustache, wearing a dark hat and a dark coat.",
20      "clothing": "Dark hat, dark coat, dark pants, and boots.",
21      "facial_expression": "Cannot be determined as his back is to the camera."
22    }
23  ],
24  "object_list": [
25    "Horse",
26    "Dirt path",
27    "Wooden buildings",
28    "Corrugated metal roofs",
29    "Smoke plume",
30    "Chimney",
31    "Forest",
32    "Rocks",
33    "Vegetation",
34    "Trees"
35  ],
36  "texture_details": [
37    {
38      "object": "Horse",
39      "texture": "Short, dark brown fur"
40    },
41    {
42      "object": "Dirt path",
43      "texture": "Uneven, dusty, with patches of grass"
44    },
45    {
46      "object": "Wooden buildings",
```

```

47     "texture": "Weathered wood planks"
48   },
49   {
50     "object": "Corrugated metal roofs",
51     "texture": "Rusted and dented metal sheets"
52   },
53   {
54     "object": "Rocks",
55     "texture": "Rough, jagged, with moss and lichen"
56   },
57   {
58     "object": "Vegetation",
59     "texture": "A mix of grass, shrubs, and trees"
60   }
61 ],
62 "lighting_details": "The scene is lit by diffused sunlight, suggesting an overcast sky. The light is soft and even, with
↳ no strong shadows.",
63 "color_palette": [
64   "#A0A0A0",
65   "#808080",
66   "#606060",
67   "#404040",
68   "#202020",
69   "#008000",
70   "#006400",
71   "#004000"
72 ],
73 "weather_conditions": "Overcast",
74 "environmental_effects": "Smoke plume rising from a chimney",
75 "animation_states": "Arthur Morgan is sitting upright on his horse, suggesting a calm and controlled ride.",
76 "error_log": "None",
77 "glitches": "None",
78 "player_status": {
79   "health": "n/a",
80   "equipment": "n/a",
81   "other_status": "n/a"
82 }
83 }

```

### Sample image-to-JSON conversion done by VIDEOGAMEBUNNY



```

1 {
2   "description": "The image showcases a close-up of a man in a World War II-era military uniform. He is positioned slightly
↳ to the right of the center, his gaze directed slightly upwards and to the left. His expression is serious, bordering
↳ on somber. He has short, dark hair and a prominent nose. The man's uniform is a muted green, with a noticeable dirt
↳ smudge on his left cheek. A dark green helmet with a netting pattern covers his head, partially obscuring his hair.
↳ The helmet's chin strap is visible, fastened with a metal buckle. A thick, dark green strap crosses his chest, likely
↳ part of a backpack or equipment harness. The background is a blurred depiction of a battlefield, with other soldiers
↳ in similar uniforms visible in the distance. The overall color palette is muted and desaturated, reflecting the grim
↳ atmosphere of war.",

```

```

3  "dialogue": [],
4  "on_screen_subtitle": "n/a",
5  "minimap_details": "n/a",
6  "inventory_display": "n/a",
7  "score_or_progress": "n/a",
8  "NPC_status": "n/a",
9  "event_indicators": "n/a",
10 "interaction_prompts": "n/a",
11 "game_mode": "n/a",
12 "HUD_description": "n/a",
13 "on_screen_watermark": "n/a",
14 "summary_of_ui_values": {},
15 "scene_description": "The scene depicts a soldier in the midst of a battlefield, likely during World War II, given the
↳ uniforms and equipment.",
16 "character_list": [
17   {
18     "name": "Unknown",
19     "appearance": "The man appears to be in his late twenties to early thirties. He has a rugged appearance with short,
↳ dark hair and a prominent nose.",
20     "clothing": "He is wearing a World War II-era military uniform in a muted green color. A dark green helmet with a
↳ netting pattern covers his head, and a thick, dark green strap crosses his chest, likely part of a backpack or
↳ equipment harness.",
21     "facial_expression": "His expression is serious and slightly somber, suggesting he is focused on the situation at hand."
22   }
23 ],
24 "object_list": [
25   "Military uniform",
26   "Helmet",
27   "Backpack strap",
28   "Backpack (partially visible)",
29   "Soldiers (in the background)"
30 ],
31 "texture_details": [
32   {
33     "object": "Helmet",
34     "texture": "Netting pattern"
35   },
36   {
37     "object": "Uniform",
38     "texture": "Camouflage pattern"
39   }
40 ],
41 "lighting_details": "The lighting is soft and diffused, likely due to overcast skies or fog. It casts subtle shadows on
↳ the man's face, accentuating his features.",
42 "color_palette": [
43   "#5C6B5A",
44   "#3D4A3B",
45   "#7A8A76",
46   "#B2B4A9"
47 ],
48 "weather_conditions": "The weather appears to be overcast or foggy, as evidenced by the soft lighting and lack of harsh
↳ shadows.",
49 "environmental_effects": "The background suggests a battlefield, with smoke or dust particles visible in the air.",
50 "animation_states": "The man appears to be in a still pose, likely captured in a moment of pause or reflection during the
↳ chaos of battle.",
51 "error_log": "None",
52 "glitches": "None",
53 "player_status": {
54   "health": "n/a",
55   "equipment": "n/a",
56   "other_status": "n/a"
57 }
58 }

```

## Sample image-to-JSON conversion done by VIDEOGAMEBUNNY



```
1 {
2   "description": "The image shows a male character in a video game, seemingly in the middle of a climbing or jumping action.
   ↳ He is positioned on a stone platform within a dimly lit, ancient-looking structure. The platform is part of a larger
   ↳ architectural complex, with visible stone walls and a barred gate in the background. The character is dressed in a
   ↳ sleeveless top and pants, suggesting a warrior or adventurer archetype. He is equipped with a sword sheathed on his
   ↳ back and appears to be in motion, with his body angled towards the platform's edge. The overall atmosphere is one of
   ↳ adventure and exploration, with the character's pose hinting at a moment of action or transition within the game.",
3   "dialogue": [],
4   "on_screen_subtitle": "n/a",
5   "minimap_details": "Cannot be determined",
6   "inventory_display": "Not visible",
7   "score_or_progress": "Not visible",
8   "NPC_status": "No NPCs present",
9   "event_indicators": "None visible",
10  "interaction_prompts": "None visible",
11  "game_mode": "Cannot be determined",
12  "HUD_description": "n/a",
13  "on_screen_watermark": "SHIRRAKO in the bottom right corner",
14  "summary_of_ui_values": {},
15  "scene_description": "The scene is set within a large, ancient-looking structure, possibly a temple or palace. The
   ↳ architecture suggests a Middle Eastern or Mediterranean influence. The environment is dimly lit, with the primary
   ↳ light source appearing to be natural light filtering in from an unseen opening above.",
16  "character_list": [
17    {
18      "name": "Unknown",
19      "appearance": "Male, athletic build",
20      "clothing": "Sleeveless top, pants, possibly leather or fabric",
21      "facial_expression": "Cannot be determined"
22    }
23  ],
24  "object_list": [
25    "Stone platform",
26    "Stone walls",
27    "Barred gate",
28    "Sword",
29    "Pottery jug"
30  ],
31  "texture_details": [
32    {
33      "object": "Stone walls",
34      "texture": "Rough, uneven stone with visible cracks and weathering"
35    },
36    {
37      "object": "Stone platform",
38      "texture": "Similar to the walls, but with a slightly smoother appearance"
39    }
40  ],
41  "lighting_details": "The scene is dimly lit, with the primary light source appearing to be natural light coming from an
   ↳ unseen opening above. This creates a sense of depth and highlights the textures of the stone surfaces.",
42  "color_palette": [
43    "#5C4A3A",
44    "#3B322A",
```

```
45     "#7A6D5B",
46     "#A1887F"
47 ],
48 "weather_conditions": "Cannot be determined",
49 "environmental_effects": "None visible",
50 "animation_states": "The character's pose suggests a dynamic action, either in the middle of a jump or a climbing
↳ maneuver.",
51 "error_log": "None visible",
52 "glitches": "None",
53 "player_status": {
54     "health": "Not visible",
55     "equipment": "Sword visible on back",
56     "other_status": "Not visible"
57 }
58 }
```

---