

Generating Automatic Commentary in Video Games using Large Language and Vision Language Models

GEORGIOS STOURNARAS, University of Twente, The Netherlands

This research explores the effectiveness of utilizing multimodal AI systems for generating accurate commentary in football video games by integrating visual information alongside text input. A prototype system, based on previous research, was developed to collect game data, and generate commentary using both a Large Language Model (LLM) and a Vision Language Model (VLM). The study compared the generated outputs of these models in producing commentary, analysing errors, and determining the impact of visual data on accuracy. The results indicated that while the VLM hallucinated less by fabricating fewer data and events, it exhibited a higher overall error rate compared to the LLM. Additionally, the image analysis often resulted in very simple and superficial commentary. Further analysis suggests that significant improvements are required in both model training and hardware capabilities to achieve real-time, accurate commentary generation. Future work will focus on refining model training with specific game data and enhancing prompt engineering to address identified limitations.

Additional Key Words and Phrases: Automatic commentary, video game, football, Large Language Model (LLM), Vision Language Model (VLM).

1 INTRODUCTION

Sport video games often have automatic commentary to enhance the gaming experience. This commentary contains play-by-play reporting, as well as providing background information, statistics or entertaining the audience (players in this case) during uneventful parts of the match. Initially, thousands of lines were written for voice actors to record, which required a significant amount of time and effort, and resulted in the commentary not sounding natural [14]. Newer recording techniques are now used so that snippets can be reused multiple times and in different order, but the storytelling is limited [7].

During the past two years, major advancements have been made in the field of machine learning, specifically in Large Language Models (LLMs). LLMs are a type of machine learning model based on the transformer architecture, that specialise in text generation. At their core, LLMs are text completion machines, i.e. they calculate what is the most probable following word given an input [9]. A very popular commercial example of such a system is ChatGPT. Systems like ChatGPT allow the user to enter a "prompt", like a request, a question, a task, and the system returns an answer [9]. For example, for the input prompt "Write me a short bedtime story" the output of the LLM could be "Once upon a time, in a land where stars painted the night sky...". LLMs are also non-deterministic, meaning that given the same input, the output will be different each time [10].

Given this information, generating commentary for video games could be a perfect application for LLMs. Because of the non-determi-

nistic nature of the models and the immense amount of training data these models are created with, game studios will not have to spend time creating unique scripts and announcements as the models are able to describe the same game events (goal, strike, hit, kill, etc.) in a different manner every time. In addition, when paired with a Text-to-Speech (TTS) system, game studios will have to spend less time and effort recording the voice lines as well.

However, a very prevalent issue with LLMs is that they hallucinate, fabricating events and data [25]. To address the issue of hallucination in LLM-generated commentary for video games, it's essential to explore newer, more powerful multimodal models. These models excel in handling multiple types of input, including both text and images [11][12]. Given that video games produce various outputs, such as internal statistics and visual frames, integrating these inputs could enhance the accuracy and reliability of the generated commentary.

The goal of my research is:

- Create a system that uses a pre-trained multimodal model to analyze game frames and game information from a video game.
- Use the same data with an existing system that uses a text unimodal model.
- Compare the accuracy of the commentary between the two systems.

Which will answer the following research question:

To what extent does the addition of visual information affect the accuracy of generated football video game commentary from a multimodal AI system?

2 RELATED WORK

Automatic generation of commentary in video games is not a novel task. For instance, [2] trained a deep-learning network on video footage from cricket games, enabling it to recognize objects in frames and generate simple commentary. Similarly, [3] used computer vision to detect UI game elements, triggering pre-engineered inputs to a GPT-3.5 model for generating commentary on a game of League of Legends. In another approach, [4] also used League of Legends, accessing the game's API to extract event data and generating commentary based on their own trained transformer model, comparing the results to other available models. Furthermore, [5] utilized a CNN to analyze frames from Minecraft, attempting to predict the associated commentary. Additionally, [1] integrated multiple input representations, such as RGB frames and optical flow, to model temporal relationships in videos and generate commentary.

The work of Czaplicki [6] is an attempt at using an LLM for commentary generation in a football video game. It uses an open-source 3D football game called "Google Research Football", initially developed as a platform for experimenting with self-learning AI players [8]. Czaplicki's system feeds game statistics into a GPT-3.5

TSciT 41, July 5, 2024, Enschede, The Netherlands

© 2024 University of Twente, Faculty of Electrical Engineering, Mathematics and Computer Science.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

model, which generates a script that is read by the system. For example, if the goalkeeper from Real Madrid saved the ball, the input would be “Event: Goalkeeper Martinez saves the ball for Real Madrid” and GPT3.5 would reply “And Martinez springs into action and saves the ball! The entire crowd goes wild.”

However, the system has a few shortcomings, including the generation of erroneous commentary; Czaplicki states that about 71% of the model’s errors were hallucinations. An example from his system: When there was a free kick in the middle of the field, the model generated an output: “And there will be a free kick for Real! United has to be careful because Ronaldo can turn it into a goal!”, despite that it was almost impossible to score from this position [6].

Based on my exploration of the literature, similar work on this topic usually trains a computer vision model on game frames (whether a real-life game or a video game) to:

- Recognize game events from understanding what is happening in the frame, and generate feedback.
- Recognize game UI elements and trigger pre-engineered requests to unimodal models similar to GPT-3.5.

The most common issue researchers mentioned is the amount of training data for their neural networks was insufficient (except [4]). That is where large models like LLMs have the advantage; commercial “off-the-shelf” models are trained on significantly more data than the specific models used in the previous research. Even though these models are built to perform well across a wide range of tasks rather than excelling in one specific task, most of the transformer models allow for “zero-shot” or “few-shot” training, which allows them to adapt to a task with no or very little training (in the form of extra instructions) [13].

To summarise, the automatic commentary in video games research that did not use an LLM had issues with the lack of sufficient training data, and the ones that did use an LLM had issues with hallucinations and maintaining context.

3 BACKGROUND

Before we continue, I will briefly introduce how LLMs work and some relevant terminology. LLMs are advanced neural networks trained to generate text. They are trained on extensive datasets comprising diverse language data, enabling them to understand and generate human-like text. These models are versatile and ready to use for various applications involving text input and output, such as chatbots, text completion, and more [15].

3.1 Prompts

LLMs operate through a series of structured interactions:

- (1) **Initial/Instruction Prompt:** This is the initial prompt that sets the tone and context for the conversation. For example, the instruction might be: “SYSTEM: You are a virtual assistant helping a curious user.” This prompt provides the model with a specific role or scenario to follow.
- (2) **User Prompt:** The user inputs a query or statement. This is the actual content that the user wants the model to respond to or interact with.
- (3) **Model Response:** Based on the initial instruction and the user prompt, the model generates a response. The response

aims to be coherent and relevant to the context set by the initial prompt.

A complete interaction would look something like this:

SYSTEM: “A chat between a curious human and an artificial intelligence assistant. The assistant gives helpful, detailed, and polite answers to the human’s questions.”
USER: “Explain LLMs to me”
ASSISTANT: “Sure, I’d be happy to. LLMs are a type of...”

To maintain the conversation’s context, all previous exchanges are included in subsequent prompts. This allows the model to “remember” past interactions and provide more coherent and contextually appropriate responses over time.

3.2 Parameters

When talking about specific LLM models, a number is associated with the name of the model (3B, 7B, 13B, any-numberB). This indicates the number of parameters a model has. These parameters refer to the numerous adjustable weights within the model that are fine-tuned during training to capture language patterns and semantics. These parameters play a crucial role in determining the model’s performance and accuracy. All other things being equal, generally the more parameters, the more powerful the model [27]. However, the higher the parameter count the more memory is required to run these models.

3.3 Quantisation

To address the memory usage issue, the process of quantisation reduces the precision of the weights used in the model to make it run on more hardware and faster. Weights are typically stored as 32-bit floating point numbers, but quantisation techniques allow those weights to be converted to 16, 8 or even 4-bit floating point numbers. However, this reduction in precision can affect the quality of the model’s predictions. On the other hand, using lower-precision floats reduces memory usage, enabling larger models to run on the same hardware, which can ultimately enhance prediction quality.

4 METHOD

My goal is to test how the addition of visual information affects the accuracy of the generated commentary. By incorporating visual cues, the system could better understand the context of the events it describes, potentially reducing errors and hallucinations in the generated commentary text. To determine this, I will first extract data from multiple matches of Google Research Football, then use the same data on two types of models; an LLM and a Vision Language Model (VLM), that accepts images and text as input. Finally, we compare the generated commentary output from the two models and see if the visual information made any difference.

4.1 Game System

I will base my work on Czaplicki’s system. It retrieves events from the Google Research Football engine, creates the prompts, asks a model to complete and then uses a TTS model to speak the generated commentary. The system uses Google’s Research Football engine,

which allows for two computer players to compete and grants us full access to the internal data, such as player information, ball coordinates, scores, and frames.

In the simulation, time is divided into discrete units called game ticks. After each tick, the system checks for specific events, like a Goal, Save, Out, Offside, Corner, etc.

In addition, if the system detects that no event has happened for a few ticks, it will generate a filler prompt, which provides the current game data and instructs the model to generate some filler talk.

When an event occurs, the system generates a dynamic prompt based on the event details, current score, and ball possession percentage. For example:

Score: 0-0 Time: 4' Possession: Real 43% - Manchester 57%
Event: the ball is kicked on corner by Martinez, Manchester United will have a chance.

The model then generates a commentary response, which is converted to speech using a Text-to-Speech (TTS) model.

Czaplicki's system was extended to be able to export, in addition to the dynamic prompt, a screenshot of the game event associated to it.

4.2 Models

Since I want to test image and text analysis, I need a multimodal model. For our vision model, I chose the Large Language and Visual Assistant (LLaVA) (version 1.5) [16][17], as it is open-source, free and is designed to run on consumer hardware. In addition, it ranks really well against the state-of-the-art GPT-4 model [18]. For our text-only LLM baseline, one option could be using Czaplicki's original approach, i.e. GPT-3.5. However, LLaVA and GPT-3.5 differ on many aspects (some of which are discussed below), and it would be difficult to attribute any difference in the generated output solely to the presence or absence of the visual modality.

4.3 Newer Version

The LLaVA VLM model I am using in this experiment is based on the Vicuna LLM [19] and an additional "vision encoder" (CLIP ViT-L/14 [20]). LLaVa/Vicuna were released almost a year after GPT-3.5, and while their training data is not public, it is likely different. Furthermore, the models have a different architecture, making a direct comparison difficult (i.e., if the two models generate different texts, it would be difficult to know if this effect should be attributed to the visual component or to other differences between the two models).

4.4 Parameters and Quantisation

LLaVA 1.5 has multiple parameter versions and supports original and 4 and 8-bit quantisation levels. The largest model that fits in our GPU (NVIDIA RTX 4080) is 13B with 4-bit quantisation.

The GPT-3.5 version of the old system, although available for consumers, is a 175B model with no quantisation used [21]. While I could use Vicuna 13B, which is the model LLaVA is based on, with a 4-bit quantisation, I cannot guarantee that it will be the exact same quantisation, due to the variety of quantization techniques

available. I could theoretically quantize both manually using the same technique, but this approach would demand substantial time and effort.

Based on these fundamental differences, it made sense to use LLaVA as our baseline. Its two modes, text and image+text are triggered by whether an image is included in the input. I'll be using "text" or "LLM" to refer to LLaVA's text only mode, and "image+text" or "VLM" to refer to LLaVA's image and text input mode.

4.5 Conversation History

A limitation of LLaVA (and most available VLMs), is that each conversation can include only one image. This restriction means that I cannot use the history of previous responses and events to enhance our commentary. For the tests, I will input the initial prompt, the image, and the event prompt for each event, creating a new conversation for each one.

4.6 Initial Prompts

As mentioned earlier, initial prompts set the quality of the responses for the prompt to come. LLaVA has default conversation prompts for both the text and image+text mode. The only difference is that in the image mode, it has a one-line instruction on how to handle the image tokens:

The visual content will be provided with the following format:
<image>visual content</image>.

Upon limited testing, removing that instruction does not alter the behaviour of the model. Further more, the model is already trained to recognise <image> tokens. So, to ensure a fair experiment, both modes will have the same instruction and user prompt, i.e. removing the <image> handling instruction from the image+text mode.

5 EXPERIMENT

With all the model considerations now in mind, our experiment consists of creating a dataset, using the same instructions and prompt input into the LLM and VLM respectively, and finally, manually check and annotate their output to measure the accuracy. Our game and model will be running on an Arch-based Linux distribution with a NVIDIA RTX 4080 with 16GB of VRAM. The LLaVA version used is llava-v1.5-13b with 4-bit quantisation. Both modes will be using the same instruction prompt:

You are a football commentator, who is getting information about a football match at Santiago Bernabeu between Real Madrid and Manchester United. I will give you info about the match and you will write commentary. Be truthful and concise, max 2 sentences per turn. Only add information about teams, players, or goals that I have specified. You may use as much of my input as required in the output, use what you need to sound natural. Avoid repeating yourself too often. Do not mention the commentary itself.

5.1 Dataset

I ran multiple games collecting events and the corresponding screenshots (Figure 1). The file looks like this after the first run:

```
"recording_timestamp": "0:00:22.438367",
"prompt": "Score: 0-0 Time: 4' Possession: Real 43% - Manchester 57% Event: the ball is kicked on corner by Martinez, Manchester United will have a chance",
"screenshot_id": "9abb271a-e35e-4aec-b609-1231c651bb86"
```

5.2 Generation

After collecting events from multiple games, for each event I generate two commentaries. The first is with the LLM (thus without using the game screenshot), where I input the instruction prompt and the event prompt, and I record the output. Output from the text-only mode is saved as "llava_response" in the file. The second commentary is generated with the VLM, where I provide the instruction prompt, the event and the screenshot of the event. Output is then saved as "llava_image" in the same file. Finally, for all our events I have a final dataset that looks like this:

```
"recording_timestamp": "0:00:22.438367",
"prompt": "Score: 0-0 Time: 4' Possession: Real 43% - Manchester 57% Event: the ball is kicked on corner by Martinez, Manchester United will have a chance",
"screenshot_id": "9abb271a-e35e-4aec-b609-1231c651bb86",
"llava_response": "Real Madrid is playing aggressive defense, but Manchester United is creating more scoring opportunities. The ball is kicked on corner by Martinez, and Manchester United will have a chance to score. ",
"llava_image": "The soccer game is currently tied at 0-0, with both teams actively competing for possession. The teams are spread out across the field, with some players closer to the center and others near the edges. The players are running and trying to gain control of the ball, which is currently located near the center of the field."
```



Fig. 1. Screenshot with id 9abb271a-e35e-4aec-b609-1231c651bb86 used as input for LLaVA. The ball is kicked out of the field (far left).

5.3 Evaluation

After generating the responses for all events, I manually annotate the outputs to check for errors and anomalies. This involves comparing the generated responses to the ground truth: the system-generated prompt and the screenshots.

6 RESULTS

I ran 7 football games, for a total of 136 events with 272 generated commentaries.

6.1 Annotation Scheme

When evaluating responses from an LLM, the evaluation is typically nuanced, as there is no binary distinction between a generated answer being "good" or "bad". Therefore, it is important to categorise the types of errors introduced or limited by the addition of images. To do this, I came up with 6 labels to annotate errors, aiming to crate them as mutually exclusive as possible. Nevertheless, there is some slight overlap between them as discussed below.

1. **Breaking Character:** When the model "breaks the 4th wall" revealing in their response the nature of the video game, the input, the model itself, or by not adhering the persona it was instructed to mimic. For example, the model was instructed to be a football sportscaster/commentator and it begins the generated response with: "In the virtual soccer game...", "In the input data provided...".
2. **Error:** Misinterpreting the input data. For example, the data in the input states that the score for teams A-B is 2-0. The response generated: "B is on the lead with the score being 2-0"
3. **General Fact:** Stating something that is generally true given our football context, but cannot be derived from the input data. Statements like "The players are tired in the second half," without specific data to support it.
4. **Hallucination:** Fabricating events and/or data. If the model responds to an offside event with "and the referee is assessing the offside" when there is no referee in the game and the match has already continued.
5. **Not mentioned:** Forgetting to mention the event that was supposed to be commented on. The input being "Score: 1-0, time: 27', Event: Ronaldo just scored a goal for Real Madrid." and the response "The match is currently at the 27th minute, with Real Madrid being in the lead."
6. **Obvious:** Making an obvious statement, like comments on the nature or the setup of the match or sport. Such as "the players are playing on a field" or "the players are wearing their teams' respective uniforms".

The labels with some overlap are 2 (Error), 3 (General Fact), and 4 (Hallucination). Given that this application is aimed at a real-time football video game, some statements are more tolerable than others. For instance, if the model states that the crowd is getting restless because there have been no goals by the 50th minute, it can be considered an acceptable statement even without crowd data. However, making up non-existent data and actions is not.

Each response can have multiple labels, as a response can have errors and be breaking character.

6.2 Results and Analysis

Error Type	Errors per mode	
	Text	Image+Text
1. Breaking Character	4	23
2. Error	18	35
3. General Fact	11	8
4. Hallucination	17	5
5. Not Mentioned	0	29
6. Obvious	0	36

Table 1. Annotated Data

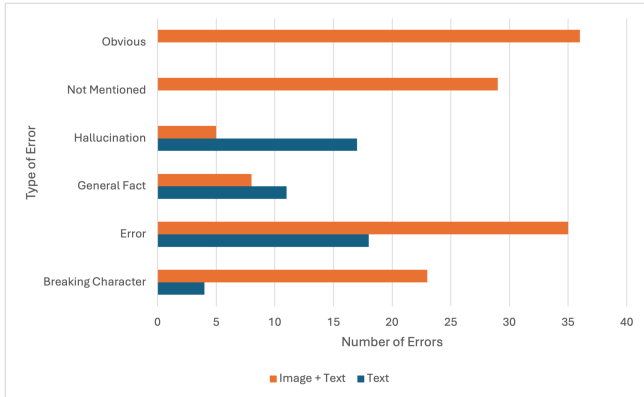


Fig. 2. Type vs Number of Errors for Text and Image+Text modes

From the results (Table 1 and Figure 2), the image+text mode made almost double the amount of errors and broke character almost 6x as much. Common errors were that both models misinterpreted the possession metrics, by saying that a team had more possession and then displaying a percentage lower than 50%. The image model would quite frequently say that both teams had equal possession of <50%, which does not add up to 100. Both models would also flip the scores, saying it was 2-0 instead of 0-2 for example.

The image+text model broke character frequently. It did so by using phrases like “in the image”, “in the virtual soccer match”, or by making comments on the match as if it was not a football commentator:

<p>Input: "prompt": "Score: 0-0 Time: 44' Possession: Real 56% - Manchester 44% Give some brief filler talk about the situation in the match"</p>
<p>Output of VLM: The players are spread across the field, with some closer to the goal and others further away. The virtual soccer game is being played on a green field with a white boundary line."</p>

The text model provided more general facts about the match, such as "fans are restless" or "Player X feels Y" after an offside. This might be because the text mode has more tokens available for responses compared to the image mode, which also needs to analyze and

comment on visuals. Each model, including ours, has a limit on the number of tokens it can generate in its output. A token can be a letter, a syllable, or a word, depending on the model [28].

I was not as harsh with labelling general facts as long as they did not classify as hallucinations.

Interestingly, the text model hallucinated a lot more than the image one, almost 3x as much. It is not clear if this is because the image helped ground the image+text mode, or because again, there was more room for tokens and the probability of fabricating events was higher. In this example, the model added that the referee missed the call when there is no referee in our video game and the offside was caught:

<p>Input: "prompt": "Score: 0-0 Time: 10' Possession: Real 19% - Manchester 81% Event: Eriksen from Manchester United is on offside"</p>
<p>Output of LLM: "Eriksen is offside, but the play continues as the referee missed the call. Real Madrid is struggling to keep possession of the ball."</p>

For our last two labels (Not Mentioned, Obvious), they were exclusive to the image one. The image model kept not mentioning the events while providing a summary of the match:

<p>Input: "prompt": "Score: 0-0 Time: 21' Possession: Real 32% - Manchester 68% Event: the ball was kicked out from the field by Rashford (Manchester United)"</p>
<p>Output of VLM: The soccer match between Real Madrid and Manchester United is currently at a 0-0 score, with both teams actively competing for the ball.</p>

When it came to image analysis, its comments were very obvious. What classifies as obvious depends on the context, but comments like “the players are on a green field” or “white lines mark the areas on the field” or “some players are closer to the ball and others further away” are not appropriate for a professional sportscaster’s commentary. While they are valid image analysis statements, they do not provide any additional insight.

An interesting finding is that almost all obvious responses also did not mention the event at hand, which might be due to (again) a token limit issue. The models are instructed to use a few sentences, and we have a few sentences about the score and time, the image and the event, so maybe the image mode event prioritised image analysis to event commentary.

Most of games had many filler events (at least 5 per game). The prompts for the filler events were originally designed with the conversation history turned on in mind, so that using previous events the model would generate a summary of the match so far.

<p>"prompt": "Score: 0-0 Time: 16' Possession: Real 21% - Manchester 79% Give some brief filler talk about the situation in the match"</p>
--

However, since I turned it off due to the LLaVA limitations, the models have very limited data to use for filler talk. The text model used mostly general facts, and in most of the cases the same facts. For example, saying "it's important for (losing team) to stay focused and look for opportunities to counterattack and create scoring chances.", or, "but neither has been able to break through the opposing team's defense yet". The image one used a lot less general facts, but did not generate any meaningful insight, mainly stating the current game status. Therefore, I did not count most of the general facts in the filler talk responses. However, errors in the scoring, possession and hallucinations were still counted.

7 DISCUSSION

RQ: To what extent does the addition of visual information affect the accuracy of generated football video game commentary from a multimodal AI system?

The addition of visual information in my experimental setup appears to negatively affect the quality of the generated commentary. It introduces more errors in interpreting the input data (events), and the additional commentary generated from the image is superficial at best.

7.1 Limitations

While the setup limitations were mentioned throughout the method section, they will be summarised here for clarity.

7.1.1 Model. The reason for the image analysis producing obvious statements like "the players are on a field" could be a model limitation. Although LLaVA showcases impressive image understanding results in its paper, it might not have been trained to understand more detailed or context-specific information beyond identifying the scene as a football game.

7.1.2 Quantisation. Both modes were not running in their original setup; they were quantised to 4 bits to fit in our VRAM. Although I am using a large model (13B), this quantisation could impact the quality of the responses.

7.1.3 Event timing and frame capture. Some screenshots for some events are a bit early or late. The issue arises because we have both a physics engine and a graphical engine running, each with its own internal ticks. With the current setup, each game engine tick equals ten physics ticks. To capture the screenshots as close to the actual event as possible, I save the previous frame's pixel array with every tick and using that instead of the frame when the event is recognised in the next tick. In this setup, there is roughly a one-frame delay between the event being rendered and it being recognised and announced. To achieve more precision in our frames, I would need to change the sampling rate from the physics engine, which would require recalculating which frame should be saved. Adjusting the sampling would require recalculating and saving all previous frames to access them when needed.

7.1.4 Initial Prompts. One major limitation might be using equal initial prompts. The image mode could have benefited from more instructions on how to handle the image, for example using "one/few shot" prompting. However, informal testing showed that additional

instructions did not significantly improve responses, suggesting that the model might not be capable of understanding more complex details from the image.

In addition, the text mode could also benefit from better instructions in the initial prompt. While the one used in the experiment generally follows published guidelines [22][23], it is likely not an optimal prompt, and would benefit from further prompt engineering.

7.1.5 Conversation History. Not having chat history is another significant limitation. The text mode could produce more meaningful summaries, comment on previous events, and avoid repeating itself. The image model could use the processed images to augment its output effectively.

7.1.6 Error Classification. One weak point of this research was the sort-of "emergent coding" used to classify errors. This method was not strictly proper emergent coding, as it was not mutually exclusive and maintaining consistency in labeling was challenging in specific contexts. As mentioned earlier, in the commentary generated for the filler prompts, I tended to be less stringent with error labeling. This is due to the models operating without memory (conversation history) of previous events, thereby having limited context to work with, and having a higher probability of stating generic football match facts.

7.2 Future Work

7.2.1 Models. Newer, more powerful models are continuously being released. For example, the authors of LLaVA have released LLaVA NeXT Video [26], a model that can understand video. OpenAI has released GPT-4o [24], which can understand video input and generate speech in (almost) real-time. Such a model could run alongside the video game to produce commentary. Existing models like GPT-4 allow for the input of multiple images in a conversation, although the API calls require paid tokens.

7.2.2 Input. An extension to this system could involve providing multiple consecutive frames for analysis instead of a single image, allowing for the extraction of temporal data and leading to more meaningful commentary. Another potential improvement is using video or a few seconds of footage before an event as input. Additionally, annotating raw screenshots to highlight the ball and players could facilitate easier and more accurate analysis. Lastly, implementing few-shot training and/or better prompt engineering could help adjust the model to respond more appropriately to specific scenarios, thereby enhancing its overall performance.

8 CONCLUSION

This research aimed to test whether using image and text input with a Vision Language Model (VLM) could provide more accurate commentary on a football video game context compared to using just a Large Language Model (LLM). A prototype was extended and adjusted from previous research to collect game data and generate responses from the models. The results showed that while both models made errors, the vision model made more errors, and its

image analysis provided very basic commentary. Numerous limitations were identified, imposed by the experimental setup or due to hardware constraints.

To create a system with greater accuracy, a good start would be training models specifically on video game data (both visual and textual), using the same engine data and type of graphical user interface (GUI). Alternatively, existing large pre-trained models can be used with adapters.

Even if I had a more powerful model capable of understanding what is happening in a football video frame, it is almost impossible to run it in real-time on consumer hardware at present.

9 APPENDIX

During the preparation of this work, the author(s) utilized ChatGPT, Copilot, and Word (Office 365) for tasks such as generating synonyms, refining sentences, and summarizing content. After using these tools/services, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the work.

10 REFERENCES

- (1) Li, Chengxi, et al. "End-to-end let's play commentary generation using multi-modal video representations." Proceedings of the 14th International Conference on the Foundations of Digital Games, 26 Aug. 2019, <https://doi.org/10.1145/3337722.3341870>
- (2) Ghosh, Debabrata, et al. "Automated cricket commentary generation using Deep Learning." INTERNATIONAL CONFERENCE ON INTELLIGENT AND SMART COMPUTATION (ICIASC-2023), 19 Mar. 2024, <https://doi.org/10.1063/5.0198759>
- (3) Ranella, Noah and Markus Eger. "Towards Automated Video Game Commentary Using Generative AI." EXAG@AIIDE (2023).
- (4) Wang, Zihan, and Naoki Yoshinaga. "Esports Data-to-Commentary Generation on Large-Scale Data-to-Text Dataset." arXiv.org, 21 Dec. 2022, <https://arxiv.org/abs/2212.10935>
- (5) Shah, Shukan, et al. "Automated Let's Play Commentary." arXiv.org, 6 Sept. 2019, <https://arxiv.org/abs/1909.02195>
- (6) Czaplicki, Michał. "Live Commentary in a Football Video Game Generated by an AI." TScIT 39, 7 July 2023, https://essay.utwente.nl/96001/1/Czaplicki_BA_EEMCS.pdf
- (7) Lee, Greg, et al. "Sports Commentary Recommendation System (SCoReS): Machine Learning for Automated Narrative." AAAI, 2012, <https://ojs.aaai.org/index.php/AIIDE/article/download/12505/12358/16027>
- (8) "Introducing Google Research Football: A Novel Reinforcement Learning Environment." Google Research, 7 June 2019, research.google/blog/introducing-google-research-football-a-novel-reinforcement-learning-environment
- (9) Noyan, Merve. "Open-Source Text Generation & Llm Ecosystem at Hugging Face." Hugging Face – The AI Community Building the Future., July 2023, <https://huggingface.co/blog/os-llms>
- (10) Ouyang, Shuyin, et al. "The Non-Determinism of Chatgpt in Code Generation." Arxiv, Aug. 2023, <https://arxiv.org/pdf/2308.02828>
- (11) OpenAI. "GPT-4." OpenAI Blog, Mar. 2023, <https://openai.com/index/gpt-4-research>
- (12) Liu, Haotian, et al. "Visual Instruction Tuning." arXiv.Org, 11 Dec. 2023, <https://arxiv.org/abs/2304.08485>
- (13) Bergmann, Dave. "What Is Zero-Shot Learning?" IBM, 24 Jan. 2024, <https://www.ibm.com/topics/zero-shot-learning>
- (14) Treadway, Dan. "How Are Commentators Programmed Into Sports Video Games?" Sports Illustrated, 21 Aug. 2014, <https://www.si.com/extra-mustard/2014/08/21/how-are-commentators-programmed-sports-video-games>
- (15) Cloudflare. "What Is a Large Language Model (LLM)?" Cloudflare, <https://www.cloudflare.com/en-gb/learning/ai/what-is-large-language-model>. Accessed 9 June 2024.
- (16) Haotian Liu. "Liuhaotian/llava-v1.5-13b · Hugging Face." Hugging Face, <https://huggingface.co/liuhaotian/llava-v1.5-13b>
- (17) Liu, Haotian, et al. "Improved Baselines With Visual Instruction Tuning." arXiv.org, 5 Oct. 2023, <https://arxiv.org/abs/2310.03744>
- (18) LLaVA. <https://llava-vl.github.io>
- (19) Chiang, Wei-Lin, et al. "Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality." LMSYS, March 2023, <https://lmsys.org/blog/2023-03-30-vicuna>
- (20) Radford, Alec, et al. "CLIP: Connecting Text and Images." OpenAI, 5 Jan. 2021, <https://openai.com/index/clip>
- (21) "Comparing GPT-3.5 and GPT-4: A Thought Framework on When to Use Each Model." Microsoft Community, 18 Mar. 2024, <https://techcommunity.microsoft.com/t5/ai-azure-ai-services-blog/comparing-gpt-3-5-amp-gpt-4-a-thought-framework-on-when-to-use/ba-p/4088645>
- (22) "Prompt Engineering." OpenAI Platform, <https://platform.openai.com/docs/guides/prompt-engineering>
- (23) "How-to Guides: Prompting." Meta, <https://llama.meta.com/docs/how-to-guides/prompting>
- (24) "Hello GPT-4o." OpenAI, 13 May 2024, <https://openai.com/index/hello-gpt-4o>
- (25) Beutel, Gernot, et al. "Artificial Hallucination: GPT on LSD?" Critical Care, vol. 27, no. 1, Apr. 2023, doi:10.1186/s13054-023-04425-6
- (26) Di Fu Jiashi Feng Ziwei Liu Chunyuan Li, Yuanhan Zhang Bo Li, Haotian Liu, Yong Jae Lee, Liangke Gui., "LLaVA-NeXT: Improved Reasoning, OCR, and World Knowledge." LLaVA, 30 Jan. 2024, llava-vl.github.io/blog/2024-04-30-llava-next-video
- (27) Touvron, Hugo et al. "Llama 2: Open Foundation and Fine-Tuned Chat Models." ArXiv abs/2307.09288 (2023).
- (28) "Tokens and Tokenization." IBM, 23 May 2024, www.ibm.com/docs/en/watsonx/saas?topic=solutions-tokens