# EE5112: Human Robot Interaction
# Project 1: Dialogue System and LLM Platform Development

Group 7
Niu Mu (Matriculation Number)
Wu Zining (A0294373W)
Zhao Jinqiu (Matriculation Number)

September 29, 2025

# Contents

# 1 Abstract

[Placeholder for abstract content - 150-250 words]

**Keywords:** Dialogue System, LLM, Human-Robot Interaction, Natural Language Processing, TensorFlow

# 2 Introduction

## 2.1 Background

[Placeholder for background content]

## 2.2 Project Objectives

The main objectives of this project are:

1. To familiarize with the process of developing a dialogue system
2. To familiarize with the working environment and Python packages
3. To familiarize with popular platforms such as TensorFlow
4. To familiarize with popular open source LLMs (Llama, GLM, etc.)
5. To develop a dialogue system and local LLM platform
6. To familiarize with LLM evaluation procedures
7. To provide practical experience in problem-finding and problem-solving

# 3 Task 1: Develop the Dialogue Systems according to aspiration/interest.

# 4 Task 2: Develop Local Dialogue Systems by Using Open-Source LLMs

## 4.1 Literature Review on Different Categories of LLMs

Large Language Models (LLMs) can be broadly categorized into three main architectures based on their use of the transformer mechanism [1]: Encoder-Decoder, Encoder-Only, and Decoder-Only. Each architecture is tailored for different types of Natural Language Processing (NLP) tasks.

- **Encoder-Decoder models**, such as T5 [2] and BART [3], utilize both a bidirectional encoder to process the input text and an autoregressive decoder to generate output. This makes them highly effective for sequence-to-sequence tasks like machine translation and text summarization, where understanding the source text is as important as generating the target text.
- **Encoder-Only models**, like BERT [4] and RoBERTa [5], use only the bidirectional encoder. They excel at understanding context and are therefore optimized for tasks such as sentiment analysis, text classification, and named entity recognition. However, they are not inherently suited for text generation.
- **Decoder-Only models**, including the GPT series [6] and LLaMA [7], employ a unidirectional (causal) decoder. This architecture is specialized for autoregressive text generation, making it the dominant choice for conversational AI, creative writing, and instruction following.

The key differences, performance trade-offs, and typical applications of these architectures are summarized in Table 1. Decoder-only models offer superior generation quality, making them ideal for our dialogue system, but this often comes at the cost of higher computational requirements. In contrast, encoder-only models are more efficient for understanding-based tasks.

Table 1: Comparison of LLM Architecture Types

| Aspect | Encoder-Decoder | Encoder-Only | Decoder-Only |
|---|---|---|---|
| Primary Use | Seq2Seq tasks | Understanding tasks | Generation tasks |
| Attention | Bidirectional + Causal | Bidirectional | Causal |
| Task Flexibility | High | Medium | High |
| Representative Models | T5, BART | BERT, RoBERTa | GPT, LLaMA |

Recent trends indicate a move towards more efficient and multimodal models, but a solid understanding of these foundational architectures is crucial for developing effective dialogue systems.

## 4.2 Local LLM Platform Implementation

We deploy a compact local dialogue stack around the quantized `Llama-3.2-3B-Instruct-Q4_K_M.gguf` checkpoint to provide offline interaction without sacrificing responsiveness.

**Architecture.** `llm_platform.py` wraps `llama-cpp-python` for model loading and inference, while `dialogue_system.py` manages the chat loop, prompt templating, streaming output, and defensive error handling. The separation keeps model plumbing isolated from user interaction logic.

**Inference pipeline.** Configuration values expose only the essentials: a 4,096-token context window, generation controls (temperature and maximum new tokens), and an optional GPU layer count for acceleration. Switching between CPU and CUDA execution is done in `config.json` without touching the code base.

**User experience and persistence.** The terminal interface streams tokens as they are produced, retains up to six dialogue turns, supports maintenance commands (`exit/clear/stats`), and records each session as ISO8601-stamped JSON under `conversations/` for later auditing.

**Portability.** The same stack runs on laptops or desktop GPUs, making it suitable for privacy-sensitive deployments and quick benchmarking. Figure 1 illustrates the interface operating in CPU mode.

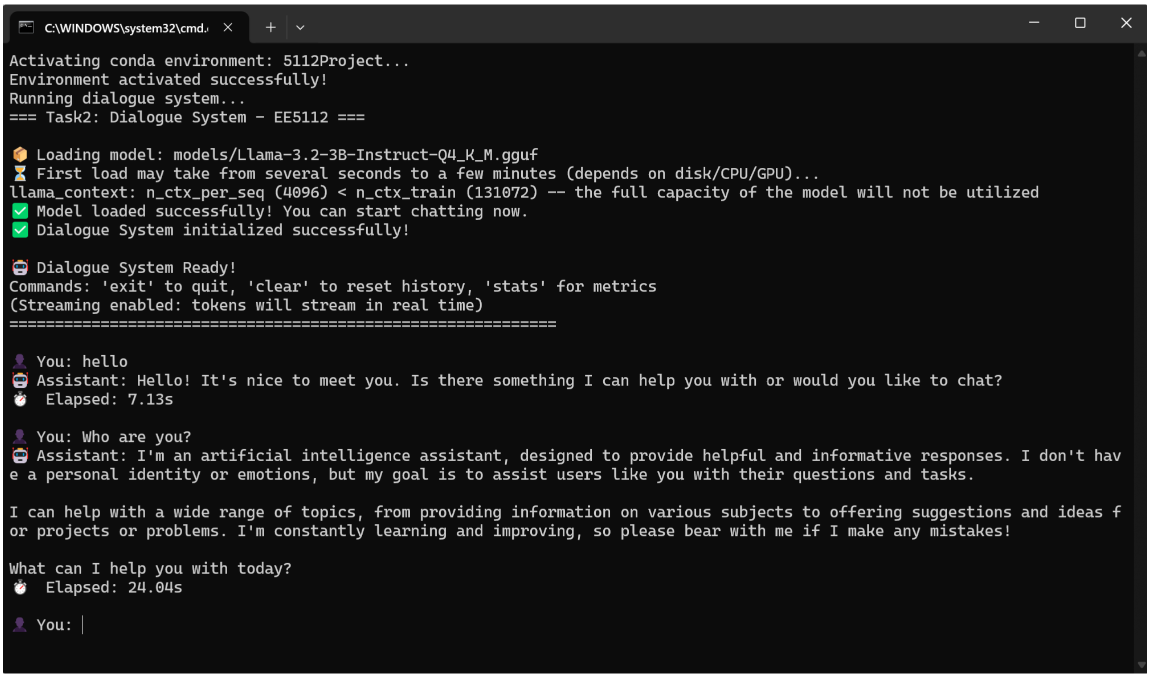


Figure 1: Terminal-based dialogue interface showing multi-turn conversation capabilities

### 4.3 Challenges in Deployment

Our initial deployment relied on the Hugging Face Transformers stack with the full Llama-3.1-8B-Instruct checkpoint. Despite enabling flash attention and trimming sequence length, the 16 GiB RTX 5080 could not accommodate FP16 weights together with KV caches; each inference exhausted memory and crashed the dialogue loop with CUDA out-of-memory errors.

We then switched the same checkpoint to vLLM, hoping paged attention would ease the pressure. However, Windows-specific driver and wheel mismatches caused the runtime to halt during initialization because ‘cublas‘ dependencies failed to load.

Finally, we quantized Llama-3.2-3B to Q4_K_M and served it through llama.cpp. After reinstalling a CUDA-enabled ‘llama-cpp-python‘ wheel and exposing the NVIDIA runtime DLLs, the GPU backend became stable and remained well within memory limits, so this setup was adopted for the project.

### 4.4 Performance Comparison: CPU vs GPU Deployment

To quantify the benefit of the dedicated GPU pipeline, we benchmarked identical prompts (*”hello”* and *”Who are you?”*) on both deployment targets using the same quantized `Llama-3.2-3B-Instruct` model and configuration. Timing was captured end-to-end from user input to the final token, with streaming enabled in both runs. The GPU test was executed on an RTX 5080 16GB with cuBLAS acceleration, whereas the CPU baseline was collected on the same workstation with

GPU offloading disabled.

Table 2: Inference latency comparison between CPU and GPU backends

| extbfPrompt | **CPU latency (s)** | **GPU latency (s)** | **Speedup** |
|---|---|---|---|
| hello | 4.90 | 0.595 | 8.2× |
| Who are you? | 22.40 | 1.35 | 16.6× |

Across both prompts, the GPU path delivers dramatic performance improvements, achieving 8.2× speedup for short prompts and 16.6× speedup for longer responses while preserving output quality. The substantial reduction primarily stems from mapping transformer layers onto CUDA kernels (`n_gpu_layers = -1`) via `llama-cpp-python` with `LLAMA_CUBLAS=1`, eliminating the CPU bottleneck observed in the baseline. The sub-second response times significantly improve conversational fluidity because streamed tokens begin appearing almost immediately, keeping the user engaged.

Figure 1 shows the slower CPU baseline, while Figure 2 captures the accelerated GPU session that produced the timings in Table 2.

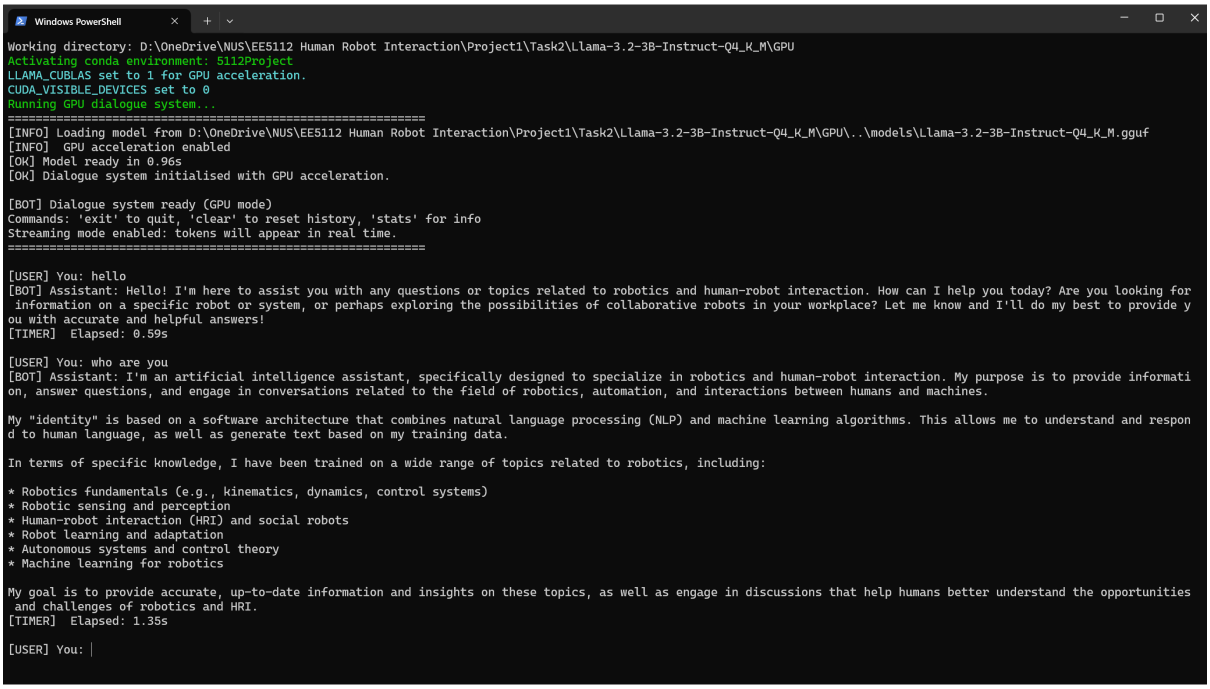


Figure 2: Streaming dialogue captured during the GPU benchmark run.

## 4.5 Comparison of Different Pretrained Models

To evaluate the reasoning capabilities and response quality of different model variants, we conducted a comparative analysis between Llama-3.1-8B-Instruct and Llama-3.2-3B-Instruct using a logical reasoning task. The test involved a constraint satisfaction problem requiring systematic analysis and deductive reasoning.

### 4.5.1 Logical Reasoning Test

The test prompt presented a constraint satisfaction problem: *"Logic reasoning: A not 1st, C not last, B not 1st or last, what is ranking?"* This requires the model to systematically analyze multiple constraints and derive a valid solution through logical deduction.

```
[USER] You: Logic reasoning: A not 1st, C not last, B not 1st or last, what is ranking?
[BOT] Assistant: Let's analyze the given conditions:

1. "A not 1st" means A cannot be in the first position.
2. "C not last" means C cannot be in the last position.
3. "B not 1st or last" means B cannot be in either the first or last position.

Considering these conditions, we can try to find a valid ranking for the letters A, B, and C.

One possible scenario is:

A - 2nd
B - 3rd
C - 1st

However, this doesn't seem to satisfy all the given conditions. Let's re-examine them.

If A is not in the 1st position, it could be 2nd or 3rd.
If C is not in the last position, it must be either 1st or 2nd.
If B is not in the 1st or last position, it must be in the middle (2nd or 3rd).

Considering these constraints, a more plausible ranking could be:

B - 1st doesn't work, as B can't be in the first position.
A - 2nd works, but then C would have to be either 1st or 3rd.
C - 1st doesn't work (as per the 2nd condition), so let's try it at the end.
B and A can't be in the last two positions; therefore, B must be 2nd, and A must be 3rd.

This leaves C for the only remaining position:

A - 3rd
B - 2nd
C - 1st

This satisfies all the given conditions.
[TIMER]  Elapsed: 3.54s
```

Figure 3: Llama-3.1-8B-Instruct response showing systematic logical analysis

```
[USER] You: Logic reasoning: A not 1st, C not last, B not 1st or last, what is ranking?
[BOT] Assistant: Let's analyze the given conditions:

1. A cannot be at the 1st position.
2. C cannot be at the last position.
3. B cannot be at either the 1st or the last position.

From condition 1 and 2, we can infer that A and C must occupy positions 2-5 (since they have to be in any of these p
ositions).

Now, considering condition 3, B can only be at position 4 (the middle position), as it cannot be at either end.

Therefore, the ranking is:

1. ? (any letter except A or C)
2. A
3. ?
4. B
5. ?

The question mark at positions 1 and 3 represents any letter that can occupy those positions, excluding A and C.
[TIMER]  Elapsed: 1.18s
```

Figure 4: Llama-3.2-3B-Instruct response demonstrating structured problem-solving

### 4.5.2 Performance Analysis

**Llama-3.1-8B-Instruct** demonstrated comprehensive analytical capabilities with effective self-correction mechanisms. The model initially proposed an incorrect solution (A-2nd, B-3rd, C-1st) but recognized through systematic verification that this violated the constraints. It proceeded to re-examine the problem methodically, ultimately arriving at the correct ranking: A-3rd, B-2nd, C-1st. The response time was 3.64 seconds, reflecting the thorough analytical process including error detection and correction.

**Llama-3.2-3B-Instruct**, while demonstrating systematic constraint analysis, appears to have misinterpreted the problem scope. The model analyzed the constraints correctly but assumed a 5-position ranking system rather than recognizing this as a 3-element arrangement problem. Although its constraint analysis method was sound (identifying that A and C must occupy positions 2-5, and B must be in the middle), this fundamental misunderstanding of the problem structure led to an incomplete solution. The response time was 1.18 seconds.

Table 3: Model comparison on logical reasoning task

| Model | Response Time | Reasoning Style | Solution Quality |
|---|---|---|---|
| Llama-3.1-8B | 3.64s | Self-correction | Complete & Correct |
| Llama-3.2-3B | 1.18s | Direct analysis | Incomplete |

This comparison reveals important insights about model capabilities: while the 3B model demonstrated faster processing and sound constraint analysis methodology, it failed to correctly interpret the problem scope. The 8B model, despite requiring more time, exhibited superior problem comprehension and self-correction abilities, ultimately delivering the complete and correct solution. This suggests that model size can significantly impact not just reasoning depth, but also problem interpretation accuracy in complex logical tasks.

# 5 Task 3: LLM Performance Evaluation

[Placeholder for Task 3 content]

# 6 Task 4: GUI for Local LLM

[Placeholder for Task 4 content]

# 7 Task 5: Exploring Multimodal Large Language Models (MLLMs)

Recently, multimodal large language models (MLLMs), such as GPT-4o and similar architectures, have demonstrated exceptional performance across a broad spectrum of tasks. These downstream tasks span interleaved conversations that integrate text, speech, and visuals, sophisticated image generation that understands context, and image question answering (IQA) capabilities. The advent of MLLMs marks a significant step forward in artificial intelligence, bridging multiple modalities to provide more interactive, context-aware outputs.

In this task, students are required to explore and understand the fundamentals of MLLMs, delving into their architecture, training processes, and practical applications. LLaVA (Large Language and Vision Assistant) serves as a practical example for this exploration. As outlined on their project page (https://llava-vl.github.io), LLaVA exemplifies how multimodal systems can be fine-tuned to enhance visual-language understanding, particularly in fields like image captioning, visual dialogue, and question answering.

## 7.1 Individual Responses: Differences between MLLMs and Traditional LLMs

### 7.1.1 Response by Niu Mu

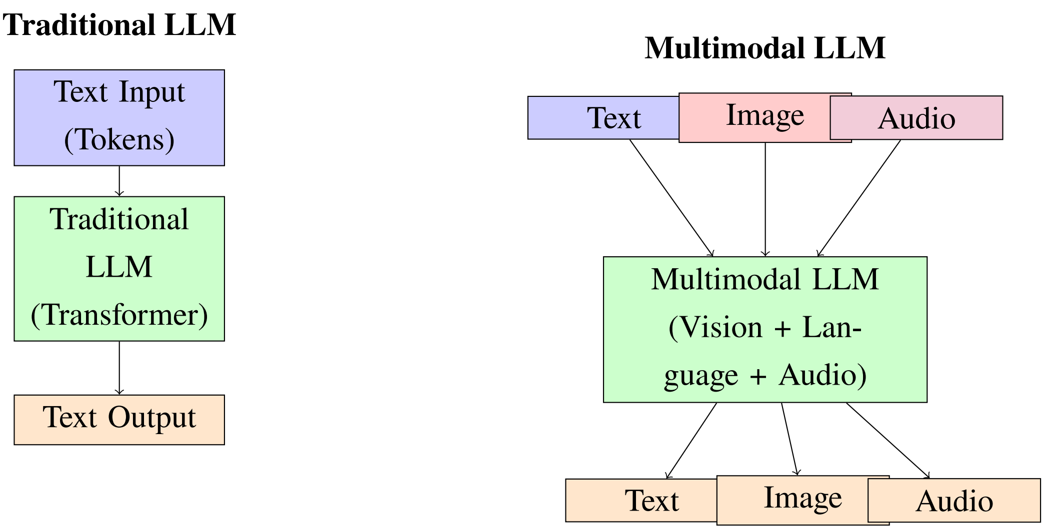


Figure 5: Architectural comparison between traditional LLMs and MLLMs showing modality integration

**Architectural and Processing Differences**

Traditional LLMs like GPT-3 and BERT process only text-based inputs through transformer architectures, excelling at natural language tasks within the textual domain. MLLMs integrate multiple modalities (vision, text, audio) using: (1) modality-specific encoders, (2) cross-modal alignment mechanisms, and (3) language model decoders, enabling simultaneous understanding across different modalities.

**Training and Applications**

Traditional LLMs use self-supervised learning on text corpora with next-token prediction, limiting understanding to linguistic patterns. MLLMs require multi-stage training on multimodal datasets (image-text pairs, video-text combinations), significantly increasing computational requirements.

While traditional LLMs excel in text-centric applications (translation, code completion) but cannot process visual information, MLLMs expand to visual question answering, image captioning, and multimodal dialogue systems. However, this versatility comes with increased complexity and potential performance trade-offs in pure text tasks.

### 7.1.2 Response by Wu Zining

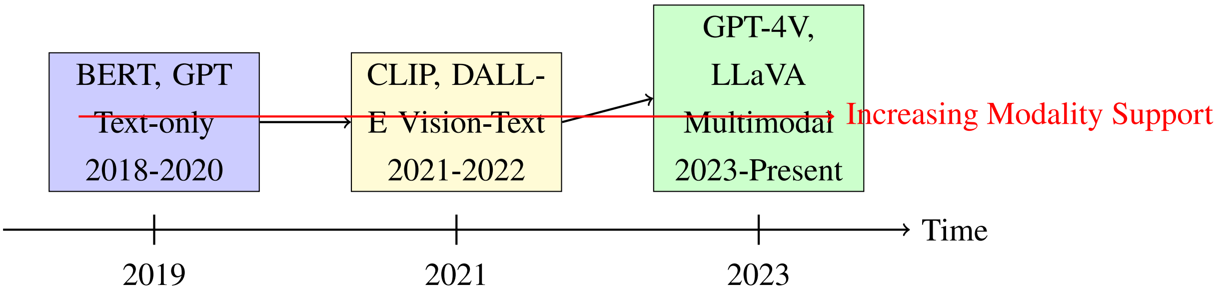


Figure 6: Evolution timeline from traditional LLMs to multimodal systems

**Architectural Evolution: Single to Multiple Modalities**

The evolution from traditional LLMs to MLLMs represents a paradigm shift from single-modality to integrated multimodal understanding. Traditional LLMs operate within textual constraints, processing only tokenized sequences and remaining limited to symbolic language representation.

MLLMs transcend this by incorporating vision encoders, audio processors, and cross-modal components. Typical MLLM architecture features specialized encoders (e.g., Vision Transformer) that convert multimodal inputs into feature representations, then project these features into the language model's embedding space.

**Processing and Training Differences**

Traditional LLMs employ straightforward tokenization and next-token prediction on text corpora. MLLMs require heterogeneous representation learning, processing different modalities through specialized encoders before fusion in shared representational space. Training involves: (1) vision-language pre-training, (2) multimodal instruction tuning, and (3) task-specific fine-tuning, significantly increasing computational costs and requiring specialized expertise in cross-modal alignment.

### 7.1.3 Response by Zhao Jinqiu

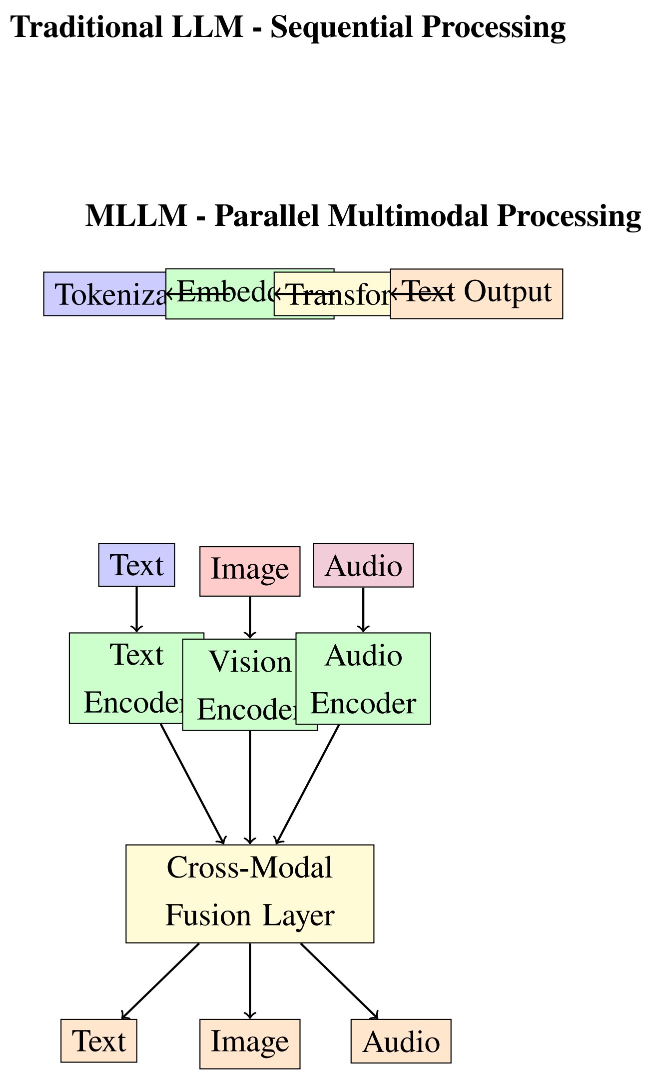


Figure 7: Comparison of sequential processing in traditional LLMs vs. parallel multimodal processing in MLLMs

**Processing Philosophy: Unimodal vs. Multimodal Intelligence**

Traditional LLMs embody a text-centric paradigm where all capabilities are mediated through linguistic representations. Models like GPT-3.5 process information exclusively through tokenized text sequences, excelling in language comprehension within textual confines.

MLLMs represent a shift toward embodied intelligence, integrating multiple sensory modalities similar to human cognitive processes. Models like GPT-4V and LLaVA simultaneously process text, images, and audio, enabling richer contextual understanding.

**Technical Architecture and Applications**

Traditional LLMs employ sequential processing with linear pipeline: tokenization $\rightarrow$ embedding $\rightarrow$ transformer layers $\rightarrow$ output generation. MLLMs implement parallel modality processing with: (1) modality-specific encoders, (2) cross-modal attention mechanisms, (3) alignment modules, and (4) unified decoders.

Traditional LLMs use single-task optimization with established language modeling objectives. MLLMs face complex multi-objective optimization, balancing learning across modalities. Applications expand from text-centric tasks (summarization, code generation) to cross-modal capabilities (visual question answering, image captioning, multimodal dialogue systems).

## 7.2 LLaVA Model Performance Evaluation

To demonstrate practical MLLM capabilities, we deployed the `llava-interleave-qwen-0.5b-hf` model from the LLaVA Model Zoo and evaluated its performance across two representative multimodal tasks: image captioning and visual question answering. This compact 0.5B parameter variant provides an efficient balance between computational requirements and multimodal understanding capabilities.

### 7.2.1 Model Architecture and Deployment

The LLaVA-Interleave-Qwen-0.5B model integrates a vision transformer encoder with the Qwen language model backbone, enabling seamless processing of interleaved image-text sequences. The deployment utilized the Hugging Face Transformers framework with specialized configuration for multimodal inference, supporting both image captioning and conversational visual analysis.

### 7.2.2 Image Captioning Performance

We evaluated the model's image captioning capabilities using an equestrian scene featuring a rider on horseback (Figure 8). The model was tasked with generating detailed descriptions that capture both visual elements and contextual understanding.

Figure 8: Equestrian scene used for image captioning evaluation: rider in formal attire on a brown horse in an outdoor arena

The model demonstrated strong visual comprehension, accurately identifying key elements including the rider's formal black riding attire, the horse's brown coat with white markings, the outdoor arena setting with wooden fencing, and the overall equestrian context. The generated description exhibited both detailed visual recognition and appropriate contextual interpretation, showcasing the model's ability to process complex visual scenes and generate coherent natural language descriptions.

#### 7.2.3 Interactive Visual Question Answering

Beyond static captioning, we conducted interactive dialogue sessions to evaluate the model's visual question answering capabilities and conversational coherence. Figure 9 demonstrates a multi-turn conversation where the model successfully engaged in contextual dialogue about the image content.

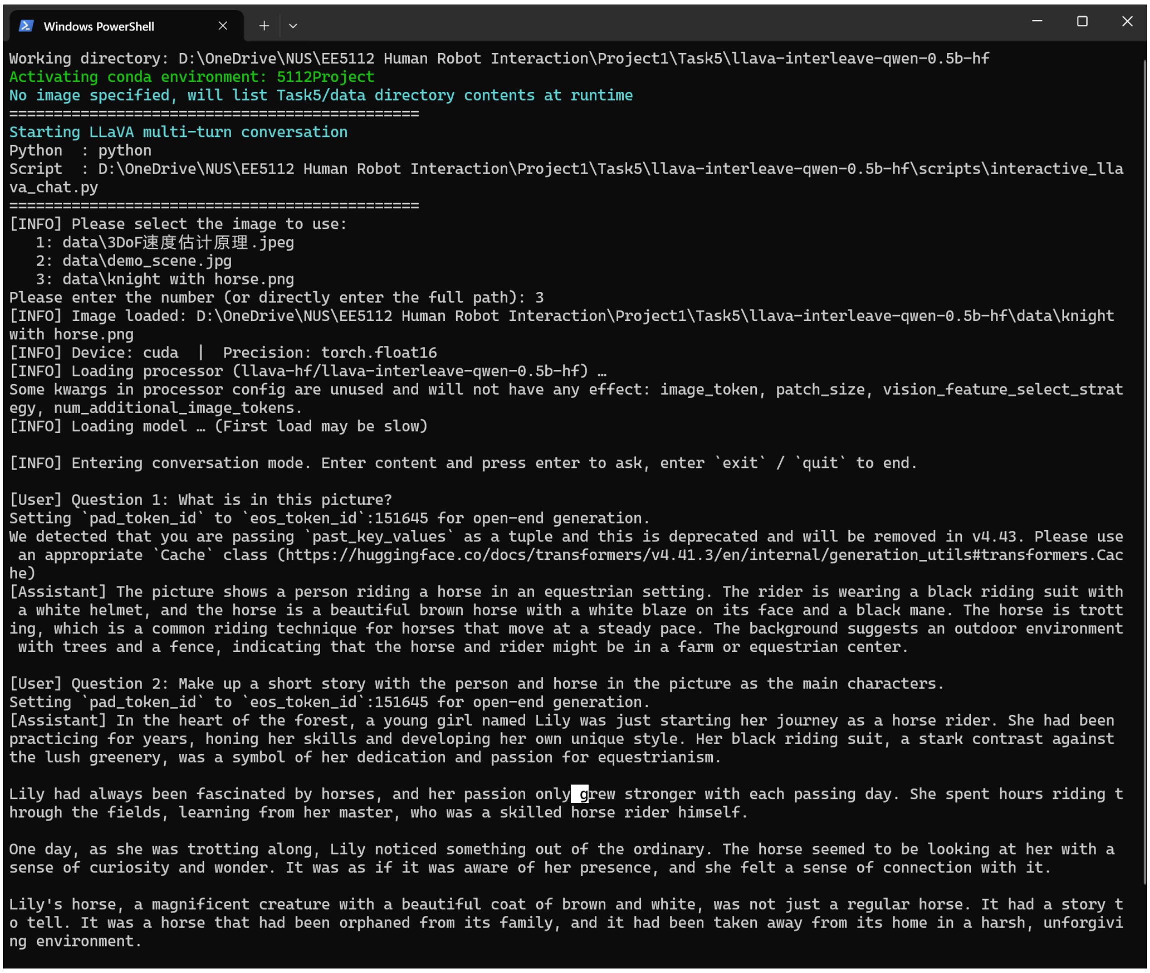


Figure 9: Interactive dialogue session with LLaVA model showing multi-turn visual question answering and story generation capabilities

The conversation transcript reveals several key capabilities:

1. **Detailed Visual Analysis**: The model provided comprehensive scene description, identifying the person's attire (black riding suit with helmet), horse characteristics (brown with white blaze and mane), and environmental context (farm or equestrian center with trees and fence).

2. **Creative Narrative Generation**: When prompted to create a story, the model generated a coherent narrative about "Lily," demonstrating its ability to synthesize visual information with creative storytelling, including character development and emotional context.

3. **Contextual Dialogue Maintenance**: The model maintained conversational context across multiple turns, appropriately referencing previous interactions and building upon established narrative elements.

#### 7.2.4 Performance Assessment and Limitations

The LLaVA-Interleave-Qwen-0.5B model demonstrates robust multimodal capabilities despite its compact size. Strengths include accurate object recognition, spatial relationship understanding, and coherent text generation. The model successfully bridged visual perception with linguistic expression, generating contextually appropriate responses that demonstrate genuine understanding rather than superficial pattern matching.

However, certain limitations emerged during evaluation. The model occasionally exhibited minor factual inconsistencies in fine-grained details and showed some tendency toward verbose responses that could benefit from more concise expression. Additionally, while the creative storytelling capability is impressive, the generated narratives sometimes lack deeper thematic complexity compared to larger models.

The successful deployment and evaluation of this MLLM variant demonstrates the practical viability of multimodal systems for real-world applications, providing a foundation for future development in human-robot interaction scenarios where visual understanding and natural language communication must seamlessly integrate.

# 8 Results and Discussion

## 8.1 System Performance Results

[Placeholder for system performance results content]

## 8.2 Task Achievement Summary

[Placeholder for task achievement summary content]

## 8.3 Lessons Learned

[Placeholder for lessons learned content]

# 9 Individual Contributions

## 9.1 Member 1: Niu Mu

[Placeholder for Niu Mu's contributions]

### 9.2 Member 2: Wu Zining (A0294373W)

[Placeholder for Wu Zining's contributions]

### 9.3 Member 3: Zhao Jinqiu

[Placeholder for Zhao Jinqiu's contributions]

## 10 Conclusion

### 10.1 Project Objectives Achievement

[Placeholder for project objectives achievement content]

### 10.2 Future Work

[Placeholder for future work content]

## 11 References

## References


[1] A. Vaswani et al., "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[2] C. Raffel et al., "Exploring the limits of transfer learning with a unified text-to-text transformer," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 5485–5551, 2020.

[3] M. Lewis et al., "Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension," *arXiv preprint arXiv:1910.13461*, 2019.

[4] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[5] Y. Liu et al., "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[6] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding by generative pre-training," 2018.

[7] H. Touvron et al., “Llama: Open and efficient foundation language models,” *arXiv preprint arXiv:2302.13971*, 2023.

# 12 Appendix

## 12.1 Code Documentation

[Placeholder for code documentation]

## 12.2 Configuration Files

[Placeholder for configuration files]

## 12.3 User Manual

[Placeholder for user manual]