MASTER THESIS

Toward a Multi Agent Approach for LLM-Based Dynamic Vehicle Control and Communication in Accidental Condition

RAFI MD ASHIFUJJMAN

STUDENT ID:
71330708

SUPERVISOR:
NAOKI FUKUTA

Department of Informatics
Graduate School of Integrated Science and Technology
Shizuoka University

SHIZUOKA UNIVERSITY

MASTER THESIS

Toward a Multi Agent Approach for LLM-Based Dynamic Vehicle Control and Communication in Accidental Condition

Author:
RAFI MD ASHIFUJJMAN

Supervisor:
Naoki FUKUTA

MASTER OF INFORMATICS

Department of Informatics
Graduate School of Integrated Science and Technology
Computer Science

July 14, 2025

Abstract

The rise of autonomous vehicles introduces a new layer of complexity to the already challenging issues of car parking. Allocating parking spaces fairly requires considering diverse perspectives, encompassing cultural norms, social values, and implicit biases. Determining a universally "fair" solution across these viewpoints remains as a significant hurdle. This research proposes a novel approach to address this challenge by investigating Large Language Models (LLMs) to evaluate fairness in parking allocation systems. LLMs, trained on massive datasets, have the potential to capture the multifaceted nature of fairness. Unlike purely mathematical approaches, they can consider various human perceptions, including cultural norms, social values, and implicit biases.

To evaluate this approach, we implement a simulation environment that generates dynamic parking scenarios. These scenarios were then fed to LLMs via Ollama and LMStudio. By analyzing the LLMs responses to these simulated parking situations, we gain insights into how people might perceive the fairness of different allocation strategies. Additionally, we conducted a comprehensive comparison of different LLM models based on various performance metrics. This research opens the door to completely new ways of allocating parking spaces for self-driving cars. Large language models (LLMs) can help us understand the people‘s consideration of fair parking allowing us to design a more user-friendly and equitable systems for autonomous transportation.

Contents

Abstract **i**

1 Introduction **1**
1.1 Research Background 1
1.2 Research Objectives 2
1.3 Research Motivation 3
1.3.1 Motivating Example 1 : What a current AV might do in Unknown situation 3
1.3.2 Motivating Example 2 : Lack of communication between vehicles 4
1.4 Research Questions 5
1.5 Research Contribution 5
1.6 Thesis Outline 6

2 Background and Related Work **7**
2.1 Autonomous Vehicle Decision-Making 7
2.2 The Challenge of Unknown-Unsafe Scenarios 8
2.3 Large Language Models (LLMs) for Reasoning 9
2.4 Multi-Agent Systems and V2V Communication 10
2.5 Related Work 10

3 Our Approach **13**
3.1 Conceptual Framework: A Multi Agent Approach 13
3.2 The Decision Agent 14
3.3 The Communication Agent 16
3.4 Text-Based Simulation and Scenario Design 17
3.5 Prompt Engineering 18

4 Conclusion and Future Work **21**
4.1 Conclusion 21

Bibliography **23**

Publications **27**

Acknowledgements **34**

5 Simulation Source Code 35
5.1 Scenario 1: Traffic Light Recognition Simulation - Autonomous Vehicle Approaching Green Traffic Light 35
5.2 Scenario 2: Stop Sign Detection Simulation - Autonomous Vehicle Approaching Stop Sign on Partial Road 37
5.3 Scenario 3: Sudden Human Crossing Detection Simulation - Autonomous Vehicle Encountering Unexpected Pedestrian 39
5.4 Scenario 4: Multi-Vehicle Collision Simulation - Autonomous Truck and Car Approaching Stop Sign with Possible Road Collision Simulation 42
5.5 Scenario 5: Multi-Agent Coordination Simulation - Three Autonomous Vehicles Coordinating Movement Around Human Obstacle . . . 45
5.6 Scenario 6: Vehicle-to-Vehicle (V2V) Communication System - Multi-Agent Information Sharing and Status Broadcasting 48
5.7 Scenario 7: LLM Decision Consistency Analysis - Statistical Evaluation of Language Model Decision Patterns in Autonomous Driving Scenarios 53

List of Figures

1.1 What a current AV might do in this situation 3
1.2 Lack of communication between vehicles may lead to an accident 4

3.1 Selected Models for the Experimentation 16
3.2 An illustration of Communication Agent 18
3.3 Illustration of real-life traffic scenarios tested in the LLM-based decision system. 19

List of Abbreviations

AI	**A**rtificial **I**ntelligence
AV	**A**utonomous **V**ehicle
ADS	**A**utonomous **D**riving **S**ystem
LLM	**L**arge **L**anguage **M**odel
MAS	**M**ulti-**A**gent **S**ystem
NLP	**N**atural **L**anguage **P**rocessing
RL	**R**einforcement **L**earning
V2V	**V**ehicle-**t**o-**V**ehicle

Chapter 1

Introduction

In the era of automation like most of ours dream is to ride on a car that will run by itself we would do nothing or can sleep in the backseat. Even if we are living in the age of AI this full automation is still a challange. In our research we are going to talk about the challanges and our approch we take to step ahead to full automation for veiches

1.1 Research Background

Our research investigates the potential of integrating Large Language Models (LLMs) [1] into a multi-agent framework [2] to improve decision making in autonomous vehicles (AVs), particularly in unknown and unsafe domains. Our main concern would be to explore the use of LLM for autonomous driving systems(ADS) [3] as the main decision-making agent within a multi-agent framework to evaluate its reasoning ability [4] to handle uncertain traffic situations. We aim to contribute for level 5 (complete automation [5]). Various standards have been defined measuring the level of AV automation, and the automotive industry usually employs a six-level classification standard ranging from 0 (fully manual) to 5 (fully autonomous) as defined by the Society of Automotive Engineers (SAE International) [6]. Our goal is to contribute to the development of safer and more reliable Level 5 [7] autonomous vehicles. In the domain of autonomous driving, traffic situations are generally categorized into three types: known-safe, known-unsafe, and unknown-unsafe scenarios. Known safe scenarios refer to routine traffic conditions in which the appropriate response is well-established and straightforward. Known unsafe situations involve clearly dangerous behaviors that are well defined and should be avoided, such as speeding in a school zone. Unknown-unsafe situations, however, are far more challenging. These are highly unpredictable events that can occur without any warning or precedent. For example, a pedestrian unexpectedly crossing the road while the light is green for oncoming traffic represents an unknown-unsafe condition. These situations are not easily foreseen and are difficult to cover in training datasets. Traditional autonomous vehicles (AVs) are generally capable of handling known-safe and known-unsafe scenarios effectively using pre-programmed rules, supervised learning, or reinforcement learning models. For instance, they can reliably detect and respond to traffic lights, lane markings, and other common features of the driving environment. However, they may have trouble facing new or unusual situations where their programmed knowledge is not directly covered in such situations. Human drivers can use common sense [8] and past experiences to handle unexpected events, such as knowing that traffic cones on a moving truck are not dangerous. However, current AV systems, even those that use rule-based approaches or reinforcement learning (RL [9]), will struggle in these scenarios.

This limitation is especially evident in what we call unknown-unsafe region situations where the correct action is not immediately clear [10]. Addressing unknown-unsafe scenarios is critical for improving AVs' ability to handle real-world driving environments, ensuring safer and more reliable autonomous systems.

1.2 Research Objectives

Determining how autonomous vehicles (AVs) should make decisions in complex or fully unknown traffic situations remains a major challenge in the development of fully autonomous systems. This problem has been widely explored in the fields of artificial intelligence, robotics, and autonomous system design. The ability to make safe and contextually appropriate decisions in real-time becomes especially critical when AVs encounter unknown-unsafe scenarios traffic conditions not represented in training data or handled poorly by traditional algorithms. Traditional autonomous vehicle systems used these approaches to take driving decisions,

- Rule-based systems depend on fixed logic and break down when encountering edge cases not covered by their rule sets.
- Supervised and machine learning-based controllers rely heavily on their training data and often fail in unfamiliar contexts, as they lack the capacity to reason beyond the patterns of learned data.

A fundamental challenge is to prepare for truly unknown situations. Any scenario researchers create is technically 'known,' even if unfamiliar to AVs, making it difficult to test how vehicles handle genuinely unknown-unsafe situations. In addition, recreating complex real-world scenarios in simulation environments is challenging, and accurately simulating unpredictable human actions or complicated environmental conditions remains nearly impossible. In real life, it is nearly impossible to include all the uncertain situations in training datasets for traditional autonomous driving systems. Human drivers rely on common sense to navigate these situations effectively. To address this limitation, we are going to investigate whether LLMs can provide human-like reasoning [11] that traditional systems lack, such as how human drivers handle

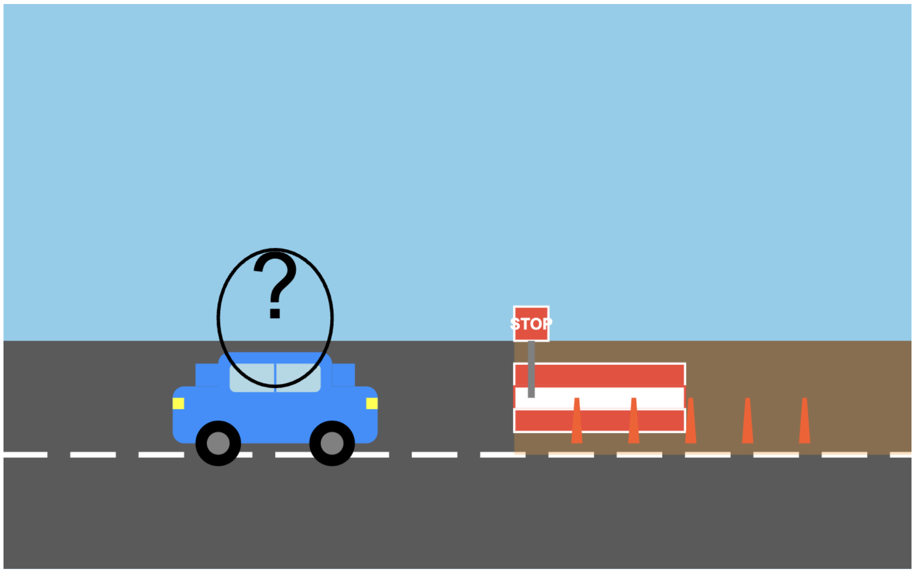


FIGURE 1.1: What a current AV might do in this situation

unexpected situations. The main objective of this research is to evaluate how effectively LLMs can act as decision agents in unfamiliar traffic conditions, and potentially bridging the gap between programmed responses and adaptive decision making in autonomous vehicle systems.

1.3 Research Motivation

The significance of unknown-unsafe scenarios lies in their unpredictability and complexity, which pose serious challenges to autonomous vehicles (AVs). Current AVs face difficulties in complex scenarios, particularly in uncertain situations where traditional pre-programmed responses can lead to issues such as, incorrectly identifying safe situations as dangerous, or failing to recognize actual dangers. Humans can use common sense and past experiences to react quickly and avoid danger. In contrast, AVs rely on pre-defined models and programmed rules, making it difficult for them to recognize the danger or respond appropriately. This limitation poses a serious safety risk in real-world driving, where unexpected situations can frequently occur. systems.

1.3.1 Motivating Example 1 : What a current AV might do in Unknown situation

Imagine a traffic situation where a road is partially blocked by construction cones, with a ‘STOP’ sign placed near the cones (Figure 1.1). A traditional AV, programmed with rigid rules, would likely interpret the ’STOP’ sign literally and come to a complete halt, potentially causing a traffic obstruction. It lacks the context to understand that the sign applies only to the blocked portion of the lane. However, a human driver would use common sense to recognize that the road is only partially closed, slow down, and safely navigate through the open section. This scenario highlights the need for a reasoning system that can interpret context beyond literal rules.

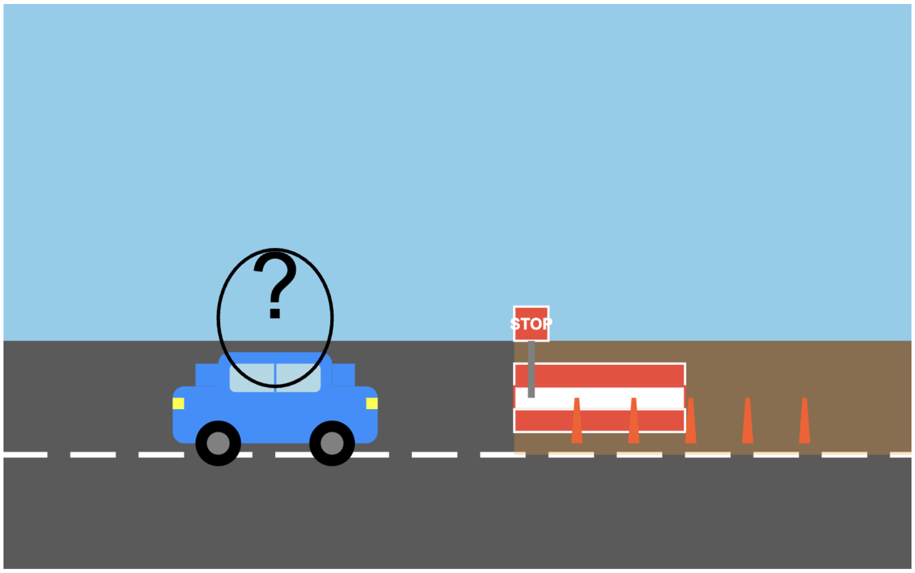


FIGURE 1.2: Lack of communication between vehicles may lead to an accident

1.3.2 Motivating Example 2 : Lack of communication between vehicles

Now, consider a more complex scenario (Figure 1.2). An autonomous truck (AV1) suddenly brakes upon seeing the ‘STOP’ sign from the previous example. Behind it is another car (AV2), which could be autonomous or human-driven. AV2’s line of sight is blocked by the truck, so it has no information about the impending stop. This can easily lead to a rear-end collision. If AV1 could communicate its intention to stop to AV2, the collision could be avoided. This illustrates a fundamental limitation of single-agent perception. The problem is not just about making the right decision for oneself, but about creating a collectively aware and safe environment through communication. Our research is motivated by the need to bridge this information gap using a multi-agent framework where agents can share their perceptions and intentions.

1.4 Research Questions

One of the primary objectives of this investigation is to answer two main research questions.

- **Research Question 1**: Can an agent make decisions with the help of an LLM without undergoing a specific learning process for each new situation?
- **Research Question 2**: Is a single centralized decision-making agent sufficient to process all incoming Vehicle-to-Vehicle (V2V) data, or is a multi-agent system required for improved efficiency and scalability?
- **Sub-Research Question**: Does repeated querying of LLMs on the same scenario lead to variations in decision-making outcomes?

1.5 Research Contribution

This research introduces a novel multi-agent architecture that integrates Large Language Models (LLMs) with Vehicle-to-Vehicle (V2V) communication [12] to enhance autonomous vehicle decision-making in unknown-unsafe driving scenarios. This work demonstrates how their combined use enables emergent reasoning capabilities, allowing autonomous vehicles to make more accurate and context-aware decisions in safety-critical scenarios. This research presents a standardized text-based simulation framework that replicates real traffic situations, used to test the LLM's capabilities in logical reasoning, rule compliance, and dynamic decision-making domain for autonomous vehicle. Additionally, a multi-agent communication system is implemented, enabling the vehicle to share structured information such as obstacle locations, speed, and future intentions in a certain radius. This data serves as a contextual input for the LLM-based Decision Agent. The work demonstrates how contextual information sharing can transform potentially hazardous scenarios into safe outcomes. Through empirical experimentation, this research provides definitive evidence that structured V2V communication data significantly improves the accuracy of LLM-based decision making. The research develops novel approaches for quantifying the consistency and reliability of decisions in LLM-based systems, addressing one of the most critical concerns for the deployment of AI in safety-critical applications. By addressing these research areas, this work contributes to the advancement of both theoretical understanding and practical implementation of AI-enhanced autonomous driving systems, with particular emphasis on navigation through safety-critical scenarios.

1.6 Thesis Outline

This thesis is organized into six different chapters as follows:

Chapter 1 ***Introduction***: This chapter describes a brief introduction to the thesis, including the research background, objectives, questions, contributions, and key motivations.

Chapter 2 ***Background and Related Work***: This chapter introduces the research topics of autonomous vehicle decision-making, unknown-unsafe scenarios, Large Language Models (LLMs), and Multi-Agent Systems, defining the basic concepts and discussing related work in the field.

Chapter 3 ***Approach and Example-based Analysis***: This chapter describes the problems and solving approach of this research and introduces the detailed multi-agent framework and example scenarios used for evaluation.

Chapter 4 ***Simulation and results***: This chapter explains the simulation setup, implementation, approach on the codes and the analysis with limitations.

Chapter 5 ***Conclusion***: This chapter describes the summary of the work, discusses the key findings, and outlines directions for future research as the conclusion.

Chapter 2

Background and Related Work

This chapter introduces the fundamental concepts and technologies that form the basis for this research. We examine the current state of autonomous vehicle decision-making systems, identify the critical challenges they face, and explore emerging technologies that can potentially address these limitations.

2.1 Autonomous Vehicle Decision-Making

The existing definitions of autonomous driving specified by the SAE for Levels 1 to 5 can be interpreted as, At level 1 vehicles, driving assistance systems that can sometimes help the driver complete some lateral or longitudinal driving tasks. At Level 2 vehicles can automatically provide multidimensional assistance. At level 3, vehicles can perform automatic acceleration and deceleration steering in a specific environment without the driver's intervention. At level 4, if a vehicle is currently in an autonomous driving state under limited conditions, the driver is not required to continuously control the steering wheel. At Level 5, vehicles can run automatically under any conditions and scenarios. At this level, the automatic system of a vehicle completely replaces the human driver and achieves complete automation. A complete autonomous vehicle (AV) is capable of sensing its environment and operating without human involvement. The decision-making process in autonomous vehicles follows a fundamental three-stage pipeline: perception, planning, and control.

- **Perception (See)**: The vehicle uses a variety of sensors and cameras, such as LiDAR (Light Detection and Ranging), and radar systems to construct a comprehensive three-dimensional representation of its environment. This stage involves detecting and classifying objects such as other vehicles, pedestrians, traffic signs, and lane markings.

- **Planning and Decision-Making (Think)**: This represents the core focus of our research. Based on the perceived environmental information, the vehicle's computational system must determine the appropriate course of action. This involves planning and taking decisions such as lane changes, speed adjustments, and navigation through complex traffic scenarios. Traditional rule-based approaches perform well in structured and predictable traffic environments. They can only execute actions that have been explicitly predefined and encoded. When unexpected or novel situations arise that fall outside the scope of their programmed rules, these systems often do not respond appropriately [13]. Our research focuses on improving the intelligence of this decision-making component through advanced reasoning capabilities.

- **Control (Act)**: Once a decision is made, the system translates the planned actions into control signals that activate the vehicle's physical components, including steering, acceleration, and braking systems. The decision-making module is particularly critical, as it is responsible for planning and performing safely.

2.2 The Challenge of Unknown-Unsafe Scenarios

The driving domain can be divided into three primary categories: known-safe, known-unsafe, and unknown-unsafe. Traditional AV systems excel in the first two categories because they can be addressed with pre-programmed rules. However, their primary limitation lies in handling unknown-unsafe scenarios.

- **Known-Safe Scenarios:** These are situations where the environment is predictable and the correct action is clear. For example, driving straight on an empty highway or stopping at a red light. These scenarios have clear rules that can be programmed into an AV's system.
- **Known-Unsafe Scenarios:** These are situations where a clear and present danger is identifiable based on established rules. Examples include an object stationary on the road or a vehicle running a red light. AVs can be trained to recognize and react to these predictable hazards.
- **Unknown-Unsafe Scenarios:** These are unexpected and unplanned situations that are not explicitly covered by pre-programmed rules. For example, a ball rolling onto the road, a pedestrian suddenly crossing outside a crosswalk, or ambiguous signals from another driver. These situations require quick and context-aware judgment.

Human drivers rely on common sense, intuition, and past experiences to navigate these unpredictable scenarios. In contrast, AVs, which depend on their pre-defined models, often struggle to recognize the nuanced danger or respond appropriately. This limitation poses a serious safety risk in real-world driving. This can manifest itself in several ways:

- **False Positives:** The AV incorrectly identifies a safe situation as dangerous, leading to unnecessary and potentially disruptive actions such as sudden braking. For instance, an AV might stop for a plastic bag blowing across the road, treating it as a solid obstacle.

- **False Negatives:** The AV fails to recognize an actual danger because it does not fit a pre-programmed pattern. This is the most critical failure mode, as it can lead directly to collisions.
- **Decision Paralysis:** Faced with an unprecedented situation, the system may not be able to make a timely decision, defaulting to a minimal risk maneuver (such as stopping) that may not be the safest overall action.

Addressing this gap is essential for the development of fully autonomous systems that can be trusted to operate safely and reliably in the complexity of real-world environments.

2.3 Large Language Models (LLMs) for Reasoning

An important development in artificial intelligence is represented by Large Language Models (LLMs). LLMs are a type of Machine Learning (ML) model trained in vast amounts of text and code from the Internet, allowing them to comprehend and produce language similar to that of a human. LLMs represent a significant leap in natural language processing and artificial intelligence. Unlike traditional rule-based systems, these machine learning models, driven by deep learning techniques, excel at comprehending and generating extensive human-like text. Reasoning is a fundamental aspect of human intelligence, essential for problem solving, decision-making, and critical thinking. In recent years, Large Language Models (LLMs) have demonstrated emerging abilities [14], such as in-context learning [15], role play [16], and analogical reasoning [17]. These abilities allow LLMs to go beyond natural language processing problems to facilitate a wider range of tasks, such as code generation [18], robotic control [19], and autonomous agents [20]. Among these abilities, human-like reasoning has garnered significant attention from both academia and industry, since it demonstrates great potential for LLMs to generalize to complex real-world problems through abstract and logical reasoning. A notable breakthrough in this area is the ‘chain of thought’ prompting technique [21], which can obtain step-by-step human-like reasoning processes at the test time without additional training.

This research investigates the potential of LLMs to serve as the main decision makers within AV systems, assessing whether they can reason and react more like humans in complex scenarios.

2.4 Multi-Agent Systems and V2V Communication

A Multi-Agent System (MAS) is a computerized system composed of multiple interacting intelligent agents. Within an MAS, each agent has an incomplete perspective and is capable of autonomous action, but the overall goal of the system is achieved through coordination and communication between agents [22]. In the context of autonomous driving, this paradigm is highly relevant. **Vehicle-to-Vehicle (V2V) communication** is a key enabling technology for a multi-agent approach to driving. V2V allows vehicles to wirelessly exchange information about their position, speed, direction, and intended actions. This creates a collective awareness that extends far beyond the perception range of any single vehicle's onboard sensors (e.g., LiDAR, cameras). By structuring a V2V system within a multi-agent framework, we can address some of the fundamental limitations of single-agent AVs: This research proposes to integrate LLM-based decision-making within a multi-agent framework, where V2V communication provides the crucial context needed for robust reasoning in accidental conditions.

2.5 Related Work

Research in autonomous vehicle decision making has evolved through several paradigms. Traditional autonomous vehicles are based on data-driven approaches, which are categorized into modular and end-to-end frameworks [23]. Modular-based systems break the entire autonomous driving process into separate components, such as the perception module, the prediction module, and the planning module. Including tasks such as object detection [24] and object occupancy prediction. Early systems relied heavily on rule-based and finite-state machines, which are effective for predictable structured environments, but lack the flexibility to handle novel situations. The work of Fu et al. [25] established the concept of 'Drive like a human', demonstrating that LLMs can generate plausible driving behaviors in response to textual descriptions of traffic scenes. Their work primarily focused on a single agent perspective and highlighted the promise of LLMs for high-level reasoning. Our research builds directly on this foundation by investigating LLM performance in a multi-agent context. The use of Multi-Agent Systems (MAS) for traffic management and coordination is well-established. Research in this area has often focused on optimizing traffic flow, coordinating intersections, and cooperative path planning [26]. However, these works typically used simpler, more formally defined agent behaviors. This research [27] presents experiments on learning decision-making policies in multi-agent environments for autonomous systems like connected autonomous vehicles. The agents were able to learn to navigate their environment and avoid collisions even in a partially observable environment with obstacles and other moving agents. However, learning decision-making policies is challenging due to the non-stationary nature of the environment. This research [28] explores a multi-agent system (MAS) architecture designed to facilitate cooperative control of CAVs. This hierarchical architecture enables vehicles to collaborate effectively in complex traffic environments, sharing information and making collective decisions that enhance safety and efficiency. The study emphasizes the importance of cooperation among autonomous vehicles, particularly in scenarios where rapid decision-making and coordination are critical to preventing accidents and ensuring smooth traffic flow. In this research [29], they explored how the combination of large language models (LLMs) with visual models, called Vision Foundation Models (VFMs), can make autonomous driving systems smarter and more capable [17]. Their work explains how technology evolved from basic sensors in early systems to more advanced deep learning methods that improve how self-driving cars see, plan, and make decisions. In this research [30], they introduced a method for training systems to better understand driving situations. They connected numerical driving data with GPT-3.5, a powerful LLM, which helped the system answer questions and make better driving decisions. The 'DriveMLM' research [31] created a new self-driving system that connects LLMs with vehicle planning tasks. This allowed the car to make decisions in a simulator and control the vehicle more effectively by using a standard planning module. The 'Drive As You Speak' [32] research took things a step further by allowing drivers to talk to the car using natural language. This made it possible for LLMs to understand and follow voice commands, improving the user experience and making driving more intuitive. In this research, AccidentGPT [33] introduced a model that can analyze traffic accidents. It uses different types of data (such as text and images) to recreate accident scenes and generate detailed reports. This research [34] reviews the integration of large language models (LLMs) into autonomous driving systems, highlighting their potential to improve decision-making, perception, and interaction through advanced reasoning and contextual understanding. The survey categorizes current research into planning, perception, question answering, and generation, addressing the challenges of transparency, scalability, and real world application. It underscores the need for robust datasets and interpretable models to build trust and improve system reliability in autonomous driving. This study [35] looks to explore the ability of integrating LLMs into autonomous driving (AD) structures to emulate human-like behavior. LLMs can use their memory to apply past experiences to future decision-making, improving adaptability and decision-making in AD systems. It can improve reliability and safety by enabling human-like reasoning and adaptability. Human drivers instinctively reason with common sense knowledge to predict hazards in unfamiliar scenarios and to understand the intentions of other road users. However, this essential capability is completely missing from traditional decision-making systems in autonomous driving. In response, this paper presents DriveLLM [36], a decision-making framework that integrates large language models (LLMs) with existing autonomous driving stacks. This integration allows for the use of common sense in decision-making. DriveLLM also features a unique cyber-physical feedback system, allowing it to learn and improve from its mistakes. In real-world case studies, the proposed framework outperforms traditional decision-making methods in complex scenarios, including difficult edge cases. However, using massive amounts of human-generated data to train these models introduces potential bias issues, as the models can unintentionally reinforce the social biases found in the data [37]. Mitigating these biases is an essential and ongoing area of research [38]. This study [39] shows a potential approach of using large language models(LLMs) to evaluate fairness.

The intersection of LLMs and multi-agent communication presents a promising new direction for autonomous driving research. Existing work has shown that LLMs can replicate human-like reasoning and that MAS frameworks improve coordination and safety. However, our research uniquely combines these strengths, and using LLMs as decision agents within a cooperative V2V-enabled architecture represents a novel integration.

Chapter 3

Our Approach

This chapter begins by introducing the overall approach and then provides a detailed explanation of our methodology, including how the experiments were set up, what assumptions were made, and the key factors considered when selecting models for evaluation. This is followed by illustrative example scenarios that demonstrate the system's behavior in representative traffic situations.

3.1 Conceptual Framework: A Multi Agent Approach

To contribute to the advancement of full automation in autonomous vehicles (AVs), this research proposes a conceptual framework based on a multi-agent architecture. The framework is designed to emulate human-like reasoning while enabling cooperative behavior among vehicles through structured communication. The framework focused on two primary agents: the Decision Agent and the Communication Agent, which work collaboratively to enhance decision - making capabilities in dynamic and uncertain traffic scenarios. The interaction between the Decision Agent and the Communication Agent is the core of this framework. The Decision Agent uses both its internal scenario understanding and externally shared V2V data to make informed decisions. This hybrid reasoning system provides the AV with both self-perception and extended perception capabilities with shared information.

- We investigated whether Large Language Models (LLMs) as a decision agent can replicate human-like decision-making processes in autonomous vehicle contexts.
- We also investigated how the information sharing facilitated by the communication agent impacts and potentially improves the accuracy decision in the autonomous driving system.

During this investigation with various LLMs as main decision agent, we proceed under two assumptions.

- 1) Certain AI systems can transform real-world situations into text-based explanations, which are later used as input for the LLMs

- 2) the outputs generated by the LLMs can be translated into actual decisions made by the decision agent by interpreting the responses from the LLMs.

We have investigated the effectiveness of various LLMs for use as decision-making agents in autonomous driving systems. Our initial objective was to answer the research question (Research Question 1): Can an agent make decisions with the help of an LLM without undergoing a specific learning process for each new situation? We have also assumed that vehicle-to-vehicle (V2V) communication has already been achieved through a standardized networking protocol, such as LAN-based direct communication or a low-latency wireless system. Under this assumption, each vehicle can communicate with other vehicles within a defined radius (e.g., 10 meters). When a vehicle enters the information sharing radius of another vehicle, it can exchange data regarding their current status, such as obstacle detection, traffic conditions, direction, and future intended actions. To evaluate the impact of V2V communication on autonomous decision-making, we have investigated how LLMs process and utilize this information. Specifically, we have examined whether the incorporation of V2V exchanged data improves decision accuracy and enables AVs to make more precise and context-aware decisions. However, V2V communication introduces a critical question (Research Question 2): Is a single centralized decision-making agent sufficient to process all incoming V2V data and make autonomous driving decisions, or is a multi-agent system required for improved efficiency and scalability? In this research, we have investigated the answers to these two research questions. Through this multi-agent approach, the research aims to bridge the gap between current autonomous vehicle capabilities and the complex decision-making requirements of fully automated transportation systems.

To test large language models as decision agent, we have prepared a set of simple, text-based simulation scenarios of different traffic situations using Python. These scenarios are designed to cover situations that require logical reasoning or the application of common sense knowledge, such as recognizing a traffic red light and stopping accordingly. These scenarios will be used as input for various LLMs to assess how effectively they handle both types of challenges. The experiments will be conducted using the LLM selected in Fig.3.

3.2 The Decision Agent

The Decision Agent is responsible for making context-aware driving decisions. It leverages the reasoning capabilities of pre-trained Large Language Models (LLMs), such as GPT-4, Claude, and open-source alternatives. These models are capable of processing natural language prompts that describe traffic scenarios and generate corresponding driving actions (e.g. STOP or FORWARD).

Hugging Face[1] is a platform where various open LLM models are available. LM Studio[2] is a platform to test and integrate LLM available on Hugging Face into a locally available system on ordinary computing devices, even without powerful GPUs. Now, **the question is what factors were considered in the selection of LLMs for this study?**

One key factor in our model selection was quantization, which optimizes model performance while reducing computational requirements. This research [40] indicates that 8-bit quantization enables the majority of LLMs to maintain a performance level comparable to their nonquantized equivalents, regardless of model size (e.g. 7B to 70B parameters). Moreover, LLMs that are quantized to 4 bits can also up hold similar performance to their non-quantized versions across most benchmarks. This approach achieves memory reduction of 50 to 75% while preserving precision in complex tasks such as reasoning, decision making, and domain-specific applications. The models chosen for this investigation were selected using a structured approach based on three key factors, **popularity and performance in text-to-text generation**, **LM Studio recommendations for optimized accuracy**, and **comparative evaluations of the top commercial services**. More specifically,

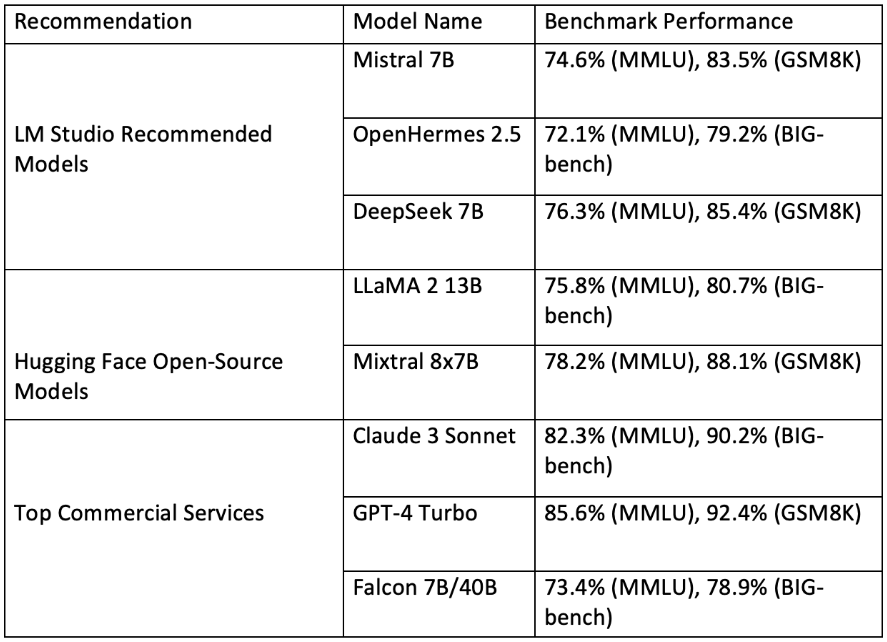
- 1) Some models were chosen based on the highest number of downloads from Hugging Face, ensuring widespread adoption and benchmark effectiveness.
- 2) Models suggested by LM Studio were included due to their strong performance and compatibility with local inference environments.
- 3) The best commercial services were selected based on their comparative performance with open-source counterparts, prioritizing accuracy, interpret ability, and real-time inference.

The selected models are shown in Figure 3.1. However, all the Hugging Face models are not fully compatible with the LM Studio runtime. As a result, some models were tested directly on the Hugging Face interface to avoid compatibility issues. Furthermore, while models such as OpenAI o1, DeepSeek R1, and Llama 3.1 405B have demonstrated strong benchmark performance, they were not included in this study due to limited quantization support, lack of local deployment feasibility, and limited open source availability.

---

[1] https://huggingface.co/

[2] https://lmstudio.ai/



Recommendation	Model Name	Benchmark Performance
LM Studio Recommended Models	Mistral 7B	74.6% (MMLU), 83.5% (GSM8K)
	OpenHermes 2.5	72.1% (MMLU), 79.2% (BIG-bench)
	DeepSeek 7B	76.3% (MMLU), 85.4% (GSM8K)
Hugging Face Open-Source Models	LLaMA 2 13B	75.8% (MMLU), 80.7% (BIG-bench)
	Mixtral 8x7B	78.2% (MMLU), 88.1% (GSM8K)
Top Commercial Services	Claude 3 Sonnet	82.3% (MMLU), 90.2% (BIG-bench)
	GPT-4 Turbo	85.6% (MMLU), 92.4% (GSM8K)
	Falcon 7B/40B	73.4% (MMLU), 78.9% (BIG-bench)

FIGURE 3.1: Selected Models for the Experimentation

3.3 The Communication Agent

The **Communication Agent** facilitates structured Vehicle-to-Vehicle (V2V) information exchange, allowing autonomous vehicles (AVs) within a defined communication radius to share and receive critical environmental data. This includes information such as obstacle presence, road conditions, current speed, and the intended action of the vehicle. To ensure compatibility with Large Language Models (LLMs), the data is formatted in a simple, human-readable structure that can be seamlessly incorporated into text-based prompts. Table 3.1 defines the standardized message schema used by the Communication Agent.

The Communication Agent ensures that all AVs within the information-sharing radius can access and benefit from collective situational awareness. For example, if a vehicle cannot directly see a stop sign due to traffic or any reason to sight block, it can still receive a warning from another vehicle that has a clear line of sight. This enables more accurate and safer decisions, particularly in complex or uncertain scenarios.

The structured data format allows for seamless integration into the LLM-based Decision Agent, allowing it to reason with both local observations and external context. Figure 3.2 illustrates the information exchange process facilitated by the Communication Agent. This multi-agent communication approach significantly improves the accuracy of the AVâĂŹs decision by expanding its environmental perception beyond the local perception limit.

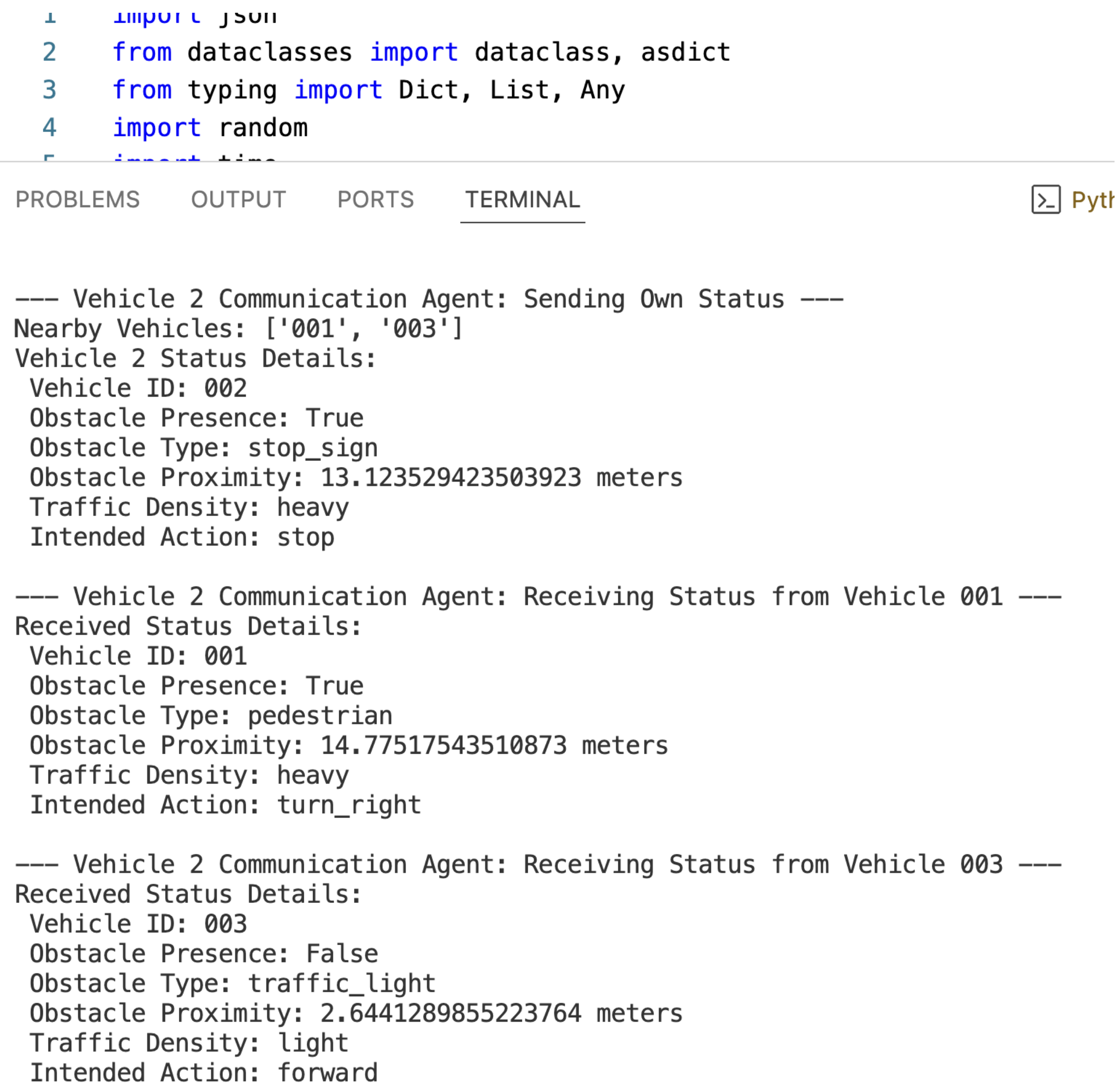
Field	**Type**	**Description**
`Vehicle_ID`	String	A unique identifier for the broadcasting vehicle.
`Current_Speed`	Float	The vehicle's current speed in meters per second (m/s).
`Obstacle_Presence`	Boolean	Indicates whether the vehicle detects a critical obstacle in its immediate path.
`Obstacle_Type`	Enum / Null	Type of detected obstacle (e.g., `'stop_sign'`, `'pedestrian'`, `'road_block'`). This is `null` if `Obstacle_Presence` is `False`.
`Obstacle_Proximity`	Float	Distance to the detected obstacle in meters.
`Intended_Action`	Enum	Planned action from the broadcasting vehicle Decision Agent (e.g., `'stop'`, `'forward'`

TABLE 3.1: Structured Information for V2V Communication

3.4 Text-Based Simulation and Scenario Design

The core concept is to investigate the reasoning capabilities of LLMs for AVs. In this research, we initially used a series of traffic situations and developed text-based simulation scenarios using Python to evaluate the effectiveness of LLMs as the decision agent in autonomous vehicle (AV). These scenarios were designed to test the capabilities of LLMs in these three domains, logical reasoning, rule compliance, and dynamic decision-making. as shown in Figure 3.3. We did extend our experiment however, capability checking domain was the same.

- **Scenario 1: Rule compliance** This scenario tested whether the LLM could correctly interpret a green traffic light and determine that the AV should proceed. It would assess the basic understanding of large language models of common traffic rules. This simulation is shown on fig 3.1(A)
- **Scenario 2: Common sense** In this case, a stop sign appears on a partially blocked road. The goal was to evaluate how the LLM interprets such ambiguous situations and whether it can determine an appropriate action based on partial cues. This simulation is shown on fig 3.1(B)
- **Scenario 3: Dynamic Decision Making** A more complex test involving a pedestrian suddenly crossing in front of the AV. This scenario assessed whether the LLM could dynamically adapt its decision to an unexpected and potentially hazardous situation. This simulation is shown on fig 3.1(C)



```python
import json
from dataclasses import dataclass, asdict
from typing import Dict, List, Any
import random
```

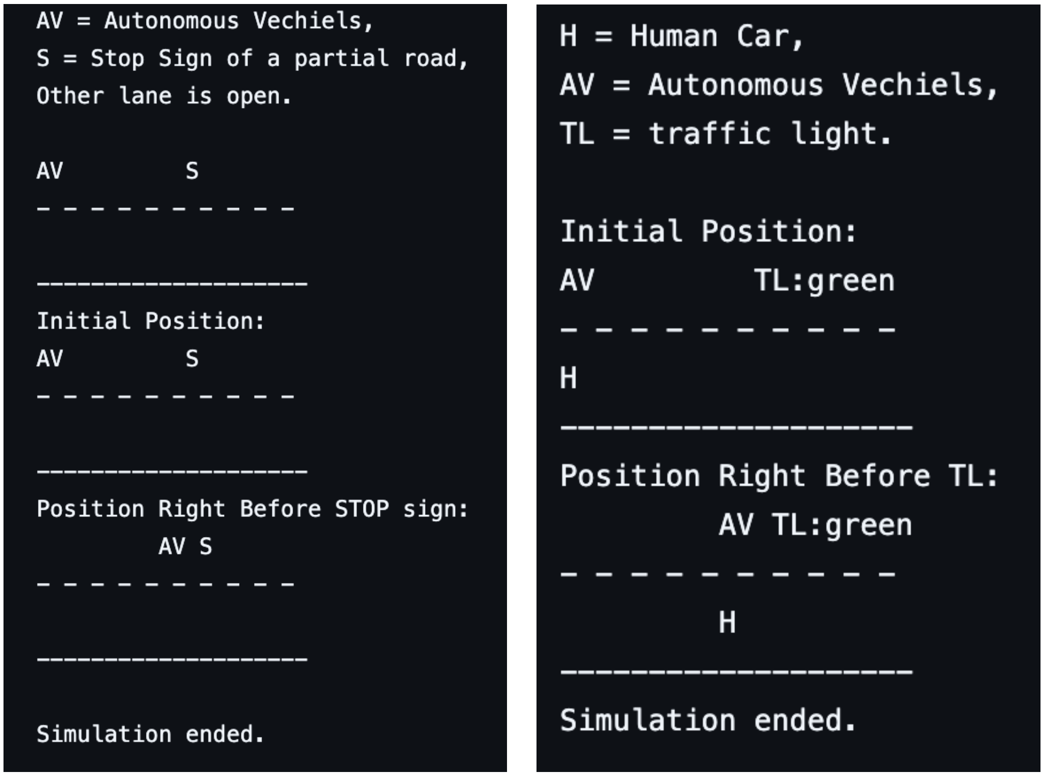
```
--- Vehicle 2 Communication Agent: Sending Own Status ---
Nearby Vehicles: ['001', '003']
Vehicle 2 Status Details:
 Vehicle ID: 002
 Obstacle Presence: True
 Obstacle Type: stop_sign
 Obstacle Proximity: 13.123529423503923 meters
 Traffic Density: heavy
 Intended Action: stop

--- Vehicle 2 Communication Agent: Receiving Status from Vehicle 001 ---
Received Status Details:
 Vehicle ID: 001
 Obstacle Presence: True
 Obstacle Type: pedestrian
 Obstacle Proximity: 14.77517543510873 meters
 Traffic Density: heavy
 Intended Action: turn_right

--- Vehicle 2 Communication Agent: Receiving Status from Vehicle 003 ---
Received Status Details:
 Vehicle ID: 003
 Obstacle Presence: False
 Obstacle Type: traffic_light
 Obstacle Proximity: 2.6441289855223764 meters
 Traffic Density: light
 Intended Action: forward
```

FIGURE 3.2: An illustration of Communication Agent

3.5 Prompt Engineering

The Natural Language Processing capabilities of Large Language Models and the growing interest in their potential applications have made prompt engineering increasingly important, as the primary method of interaction with LLMs is through prompts [41]. The design of the prompt determines LLM behavior, making prompt engineering critical, since the prompt acts as a guide and directly impacts the results. To carry out our experiment, we tested different LLM models with text-based traffic simulation scenarios using a simple prompt. Each scenario was presented to various LLM models using this simple prompt asking the models to simply respond with a single word without further explanation,

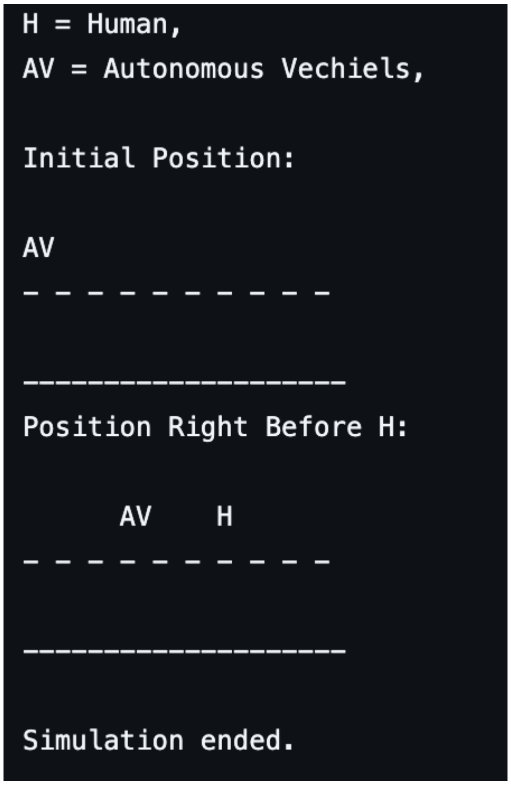
"What would the AV do? Please, just answer STOP or FORWARD."



```
AV = Autonomous Vechiels,
S = Stop Sign of a partial road,
Other lane is open.

AV         S
- - - - - - - - - -


--------------------
Initial Position:
AV         S
- - - - - - - - - -


--------------------
Position Right Before STOP sign:
         AV S
- - - - - - - - - -


--------------------


Simulation ended.
```

```
H = Human Car,
AV = Autonomous Vechiels,
TL = traffic light.

Initial Position:
AV         TL:green
- - - - - - - - - -
H
--------------------
Position Right Before TL:
         AV TL:green
- - - - - - - - - -
         H
--------------------
Simulation ended.
```

((A)) Traffic Light ((B)) Stop Sign on Partial Road



```
H = Human,
AV = Autonomous Vechiels,

Initial Position:

AV
- - - - - - - - - -


--------------------
Position Right Before H:

      AV    H
- - - - - - - - - -


--------------------


Simulation ended.
```

((C)) Human Suddenly Crossing Road

FIGURE 3.3: Illustration of real-life traffic scenarios tested in the LLM-based decision system.

We are assuming that this would be used as a real decision in the autonomous driving system.

Chapter 4

Conclusion and Future Work

4.1 Conclusion

We presented an approach for leveraging Large Language Models (LLMs) to enhance reasoning capabilities in uncertain traffic situations within a multi-agent framework for autonomous vehicles. We have investigated LLMs from Hugging Face, LM Studio, as well as major commercial services such as Claude, GPT-4, and Falcon. Their responses to basic traffic scenarios were analyzed, revealing varying levels of consistency and accuracy. Our results show that for RQ1, Large Language Models (LLMs) were able to make intelligent and safe decisions even in unfamiliar or novel traffic situations. They did not require specific training for each scenario and showed strong general reasoning skills, similar to how a human driver would respond to unexpected conditions. For RQ2, the use of multiple agents, such as a Decision Agent and a Communication Agent, improved the system performance compared to the single agent setup. When vehicles shared data such as speed, obstacles, and intentions through V2V communication, decisions became more accurate, consistent, and safer. This design also made the system more scalable and flexible for handling complex real-world traffic situations. These results reveal both the promise and the current limitations of this approach, highlighting the need for improved consistency in the model responses. The proposed approach aims at the realization of adaptive autonomous systems capable of human-like reasoning in unpredictable situations. By integrating contextual data derived from V2V communication with LLM reasoning, our goal is to bridge the critical gap in decision-making accuracy for autonomous vehicles.

Limitations and Future Work

- **Simulation Expansion:** Extend the simulation environment to include multilane roads and multiple vehicles, simulating more realistic traffic conditions with varying visibility and road complexity.

- **Increased the number of trails for Consistency Testing:** Run the same prompt more than 100 times per scenario per LLM to obtain a stronger statistical basis to evaluate the consistency of the decision.
- **Offline Evaluation of Local LLMs:** Test the effectiveness of local LLMs in disconnected environments to assess their viability in real-world AV applications.
- **Emergency Broadcasting via Communication Agent:**
  - Implement dynamic emergency messaging (e.g., broadcast `STOP` or `CAUTION` to nearby vehicles).
  - Evaluate radius-based alerts: Determine what warnings should be sent to the following vehicles based on proximity and shared intent.
- **Limited Scenario Diversity:** The simulation environment includes a limited range of traffic scenarios, which may restrict the system's ability to generalize to real-world driving conditions.
- **Lack of Fail-Safe Mechanisms:** The current system lacks built-in redundancy or error handling mechanisms to recover from incorrect LLM output or detect unsafe decisions in real time.
- **Unaddressed V2V Communication Constraints:** This study assumes ideal communication conditions, without evaluating real-world issues such as network congestion, signal interference, or latency, especially in dense traffic scenarios.

Bibliography

[1] Y. Chang, X. Wang, J. Wang, *et al.*, "A survey on evaluation of large language models," *ACM transactions on intelligent systems and technology*, vol. 15, no. 3, pp. 1–45, 2024.

[2] Y. Wang, Z. Wu, J. Yao, and J. Su, "Tdag: A multi-agent framework based on dynamic task decomposition and agent generation," *Neural Networks*, p. 107 200, 2025.

[3] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE access*, vol. 8, pp. 58 443–58 469, 2020.

[4] J. L. Espejel, E. H. Ettifouri, M. S. Y. Alassan, E. M. Chouham, and W. Dahhane, "Gpt-3.5, gpt-4, or bard? evaluating llms reasoning ability in zero-shot setting and performance boosting through prompts," *Natural Language Processing Journal*, vol. 5, p. 100 032, 2023.

[5] M. Raza, "Autonomous vehicles: Levels, technologies, impacts and concerns," *International Journal of Applied Engineering Research*, vol. 13, no. 16, pp. 12 710–12 714, 2018.

[6] T. Eley IV, J. L. King, K. Lyytinen, and J. V. Nickerson, *SAE J3016 as a Learning Device for the Driving Automation Community: Technical, Socio-technical, and Systemic Learning*. SAE International, 2024.

[7] J. Cho and J. Heo, "The more you know, the more you can trust: DriversâĂŹ understanding of the advanced driver assistance system," in *HCI in Mobility, Transport, and Automotive Systems. Automated Driving and In-Vehicle Experience Design: Second International Conference, MobiTAS 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part I 22*, Springer, 2020, pp. 230–248.

[8] C. Geertz, "Common sense as a cultural system," *The Antioch Review*, vol. 33, no. 1, pp. 5–26, 1975.

[9] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[10] D. Bogdoll, J. Breitenstein, F. Heidecker, *et al.*, "Description of corner cases in automated driving: Goals and challenges," in *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2021, pp. 1023–1028. DOI: `10.1109/ICCVW54120.2021.00119`.

[11] P. N. Johnson-Laird, "Mental models and human reasoning," *Proceedings of the National Academy of Sciences*, vol. 107, no. 43, pp. 18 243–18 250, 2010.

[12] A. Demba and D. P. Möller, "Vehicle-to-vehicle communication technology," in *2018 IEEE international conference on electro/information technology (EIT)*, IEEE, 2018, pp. 0459–0464.

[13] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018.

[14] J. Wei, Y. Tay, R. Bommasani, *et al.*, "Emergent abilities of large language models," *arXiv preprint arXiv:2206.07682*, 2022.

[15] Q. Dong, L. Li, D. Dai, *et al.*, "A survey on in-context learning," *arXiv preprint arXiv:2301.00234*, 2022.

[16] M. Shanahan, K. McDonell, and L. Reynolds, "Role play with large language models," *Nature*, vol. 623, no. 7987, pp. 493–498, 2023.

[17] L. H. M. Lam, R. K. Thatikonda, and E. Shareghi, "A closer look at tool-based logical reasoning with llms: The choice of tool matters," in *Australasian Language Technology Association Workshop*, 2024. [Online]. Available: `https://api.semanticscholar.org/CorpusID:270219176`.

[18] J. Gehring, K. Zheng, J. Copet, *et al.*, "Rlef: Grounding code llms in execution feedback with reinforcement learning," *arXiv preprint arXiv:2410.02089*, 2024.

[19] A. Brohan, Y. Chebotar, C. Finn, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," in *Conference on robot learning*, PMLR, 2023, pp. 287–318.

[20] X. Deng, Y. Gu, B. Zheng, *et al.*, "Mind2web: Towards a generalist agent for the web," *Advances in Neural Information Processing Systems*, vol. 36, pp. 28 091–28 114, 2023.

[21] J. Wei, X. Wang, D. Schuurmans, *et al.*, "Chain-of-thought prompting elicits reasoning in large language models," *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.

[22] O. F. Rana and K. Stout, "What is scalability in multi-agent systems?" In *Proceedings of the fourth international conference on Autonomous agents*, 2000, pp. 56–63.

[23] L. Chen, P. Wu, K. Chitta, B. Jaeger, A. Geiger, and H. Li, "End-to-end autonomous driving: Challenges and frontiers," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, pp. 10 164–10 183, 2023. [Online]. Available: `https://api.semanticscholar.org/CorpusID:259287283`.

[24] Y. Wang, V. C. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon, "Detr3d: 3d object detection from multi-view images via 3d-to-2d queries," *ArXiv*, vol. abs/2110.06922, 2021. [Online]. Available: `https://api.semanticscholar.org/CorpusID:238744328`.

[25] D. Fu, X. Li, L. Wen, *et al.*, "Drive like a human: Rethinking autonomous driving with large language models," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 910–919.

[26] A. Iftekhar and N. Fukuta, "Toward an efficient and resilient electric vehicle charging place allocation using multiagent approach," in *2024 16th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, 2024, pp. 663–664. DOI: `10.1109/IIAI-AAI63651.2024.00126`.

[27] X. Wang, B. Lv, K. Wang, and R. Zhang, "Asts: Autonomous switching of task-level strategies," ENG, *International Journal of Applied Mathematics and Computer Science*, vol. 33, no. 4, 553âĂŞ568, 2023.

[28] J. Liang, Y. Li, G. Yin, *et al.*, "A mas-based hierarchical architecture for the cooperation control of connected and automated vehicles," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 2, pp. 1559–1573, 2022.

[29] C. Cui, Y. Ma, X. Cao, *et al.*, "A survey on multimodal large language models for autonomous driving," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 958–979.

[30] L. Chen, O. Sinavski, J. Hünermann, *et al.*, "Driving with llms: Fusing object-level vector modality for explainable autonomous driving," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2024, pp. 14 093–14 100.

[31] W. Wang, J. Xie, C. Hu, *et al.*, "Drivemlm: Aligning multi-modal large language models with behavioral planning states for autonomous driving," *arXiv preprint arXiv:2312.09245*, 2023.

[32] C. Cui, Y. Ma, X. Cao, W. Ye, and Z. Wang, "Drive as you speak: Enabling human-like interaction with large language models in autonomous vehicles," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 902–909.

[33] L. Wang, Y. Ren, H. Jiang, *et al.*, "Accidentgpt: A v2x environmental perception multi-modal large model for accident analysis and prevention," in *2024 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2024, pp. 472–477.

[34] M. Elhenawy, A. Abutahoun, T. I. Alhadidi, *et al.*, "Visual reasoning and multi-agent approach in multimodal large language models (mllms): Solving tsp and mtsp combinatorial challenges," *arXiv preprint arXiv:2407.00092*, 2024.

[35] A. K, S. Sudarshan G S, and S. J U, "Llm's for autonomous driving: A new way to teach machines to drive," in *2023 3rd International Conference on Mobile Networks and Wireless Communications (ICMNWC)*, 2023, pp. 1–6. DOI: `10.1109/ICMNWC60182.2023.10435998`.

[36] L. Chen, O. Sinavski, J. HÃijnermann, *et al.*, "Driving with llms: Fusing object-level vector modality for explainable autonomous driving," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2024, pp. 14 093–14 100. DOI: `10.1109/ICRA57147.2024.10611018`.

[37] B. Friedman and H. Nissenbaum, "Bias in computer systems," *ACM Transactions on information systems (TOIS)*, vol. 14, no. 3, pp. 330–347, 1996.

[38] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *ACM computing surveys (CSUR)*, vol. 54, no. 6, pp. 1–35, 2021.

[39] S. S. Duddela and N. Fukuta, "Toward an acceptable dynamic allocation considering fairness on autonomous car parking," in *2024 16th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, 2024, pp. 286–291. DOI: `10.1109/IIAI-AAI63651.2024.00061`.

[40] R. Jin, J. Du, W. Huang, *et al.*, "A comprehensive evaluation of quantization strategies for large language models," *ArXiv*, vol. abs/2402.16775, 2024. [Online]. Available: `https://api.semanticscholar.org/CorpusID:268032411`.

[41] J. White, Q. Fu, S. Hays, *et al.*, "A prompt pattern catalog to enhance prompt engineering with chatgpt," *arXiv preprint arXiv:2302.11382*, 2023.

Publications

International Conference

Toward a Multi Agent Approach for LLM-Based Dynamic Vehicle Control and Communication in Accidental Condition

Rafi Md Ashifujjman
Graduate School of Integrated Science and Technology
Shizuoka University
Hamamatsu, Japan
rafi.md.ashifujjman.23@shizuoka.ac.jp

Naoki Fukuta
College of Informatics, Academic Institute
Shizuoka University
Hamamatsu, Japan
fukuta@inf.shizuoka.ac.jp

***Abstract*—In autonomous vehicles (AVs), making the right decision in unknown traffic situations remains a challenge. Traditional AVs rely on pre-trained models, which often struggle in such cases where human-like reasoning is required. In this paper, we examine how Large Language Models (LLMs) act as a decision agent responding to scenario-specific text prompts derived from real-world traffic situations. In addition, we propose a Communication Agent that enables vehicle-to-vehicle (V2V) information sharing, such as speed, obstacles, and future intentions, in a structured format to provide contextual support for the decision agent. This paper proposes a preliminary approach to assess whether LLMs within a multi-agent framework, when provided with structured prompts and contextual data, can support consistent and human-like decision making.**

***Index Terms*—autonomous vehicles, large language models (LLMs), multi-agent, vehicle-to-vehicle (V2V) communication, decision making agent, communication agent.**

I. INTRODUCTION

This paper explores the potential of integrating Large Language Models (LLMs) [1] into a multi-agent framework [2] to improve decision making in autonomous vehicles (AVs), particularly in unknown and unsafe traffic situations. Our main concern would be to explore the use of LLMs for autonomous driving systems (ADS) as the main decision-making agent within a multi-agent framework to evaluate its reasoning ability [3] to handle uncertain traffic situations. Our goal is to contribute to the development of safer and more reliable Level 5 [4] autonomous vehicles. We aim to contribute to Level 5 (complete automation [5]). There are numerous instances in which traditional autonomous vehicles are capable of making effective decisions in familiar scenarios using pre-trained models. However, they may have trouble facing new or unusual situations where their programmed knowledge is not directly covered in such situations. Human drivers can use common sense and past experiences to handle unexpected events, such as knowing that traffic cones on a moving truck are not dangerous. However, current AV systems, even those that use rule-based approaches or reinforcement learning (RL [6]), will struggle in these scenarios. This limitation is especially evident in what we call unknown-unsafe domain situations where the correct action is not immediately clear [7]. During this investigation with various LLMs, we proceed under two assumptions: 1) certain AI systems can transform real-world situations into text-based explanations, which are later used as input for the LLMs, and 2) the outputs generated by the LLMs can be translated into actual decisions made by the AV's decision agent by interpreting the responses from the LLMs. We are investigating the effectiveness of various LLMs for use as decision-making agents in autonomous driving systems. Our initial objective is to answer the research question (Research Question 1): Can an agent make decisions with the help of an LLM without undergoing a specific learning process for each new situation? In our research, we will assume that vehicle-to-vehicle (V2V) communication has been achieved through a standardized networking protocol, such as LAN-based direct communication or a low-latency wireless system. Under this assumption, each vehicle can communicate with other vehicles within a defined radius (e.g., 10 meters). When a vehicle enters the information sharing radius of another vehicle, it can exchange data regarding their current status, such as obstacle detection, traffic conditions, direction, and future intended actions. To evaluate the impact of V2V communication on autonomous decision-making, we will investigate how LLMs process and utilize this information. Specifically, we will examine whether the incorporation of V2V exchanged data improves decision accuracy and enables AVs to make more precise and context-aware decisions. However, V2V communication introduces a critical question, (Research Question 2): Is a single centralized decision-making agent sufficient to process all incoming V2V data and make autonomous driving decisions, or is a multi-agent system required for improved efficiency and scalability? In this paper, we investigate the answers to these two research questions.

II. MOTIVATION AND BACKGROUND

Imagine a traffic situation, where a road is partially blocked due to construction, with traffic cones and a 'STOP' sign indicating a partial road block. A typical autonomous vehicle (AV) might see the 'STOP' sign and interpret it as a complete

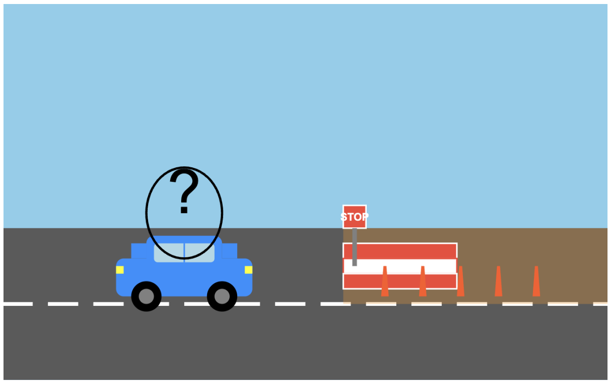


Fig. 1: What a current AV might do in this situation?

road closure, coming to a complete stop because that is what its pre-programmed data tell it to do. However, a human driver in the same situation might use common sense, realize that the road is only partially closed, and safely continue through the open section.

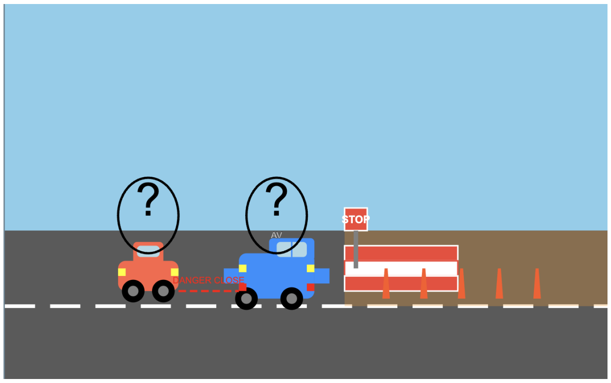


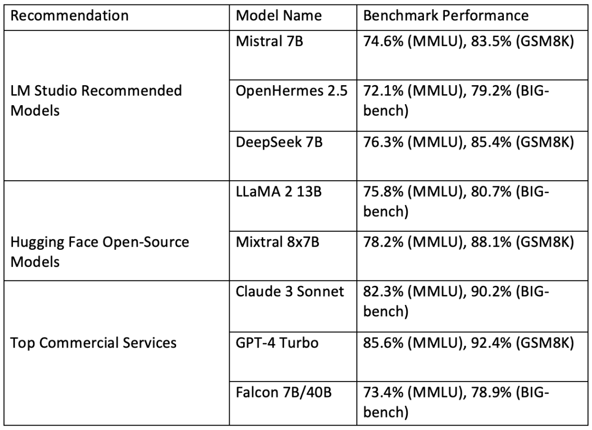
Fig. 2: Lack of communication between vehicles may lead to a potential accident.

Now imagine a more complicated situation where an autonomous truck comes to a sudden stop when it sees the 'STOP' sign. There is another car, driven by a human or another AV, right behind it. This sudden stop could easily cause a crash. Now imagine a human driver encountering a partially blocked road. To navigate safely, the driver would shift into the adjacent open lane, relying on instinct, experience, and situational awareness. Anticipating the possibility of an oncoming vehicle from the opposite direction or from behind, the driver would proactively slow down, assessing the risk, and would take the necessary precautions to avoid a collision. In contrast, a traditional AV, which lacks human-like reasoning and predictive thinking, may not recognize the potential danger. Without contextual understanding, it can continue at its normal speed, assuming the lane is clear, increasing the risk of a head-on collision.

However, if the AV had better reasoning abilities, it might recognize that the road is only partially closed and proceed safely through the open part. And if the AV could communicate with the vehicle behind it, it could alert the other driver about its intended actions, particularly if the lane were to close suddenly, helping to prevent a potential accident. A fundamental challenge in traditional autonomous vehicles (AVs) is to prepare for all truly unknown situations [8]. Any scenario we create is technically 'known' to us, even if it feels unfamiliar to the AVs, making it difficult to test how AVs handle truly unknown-unsafe situations. In addition, it is difficult to recreate complex real-world scenarios in simulation environments. For example, it is difficult to accurately simulate unpredictable human actions or complicated environmental conditions. In real life, there could be countless unexpected situations that are nearly impossible to include in training data sets for traditional autonomous driving systems (ADS) to avoid accidents. Human drivers use common sense to deal with these situations, so through this paper we will explore whether LLMs can also show human-like reasoning to handle them effectively or not.

To address these challenges, we propose a possible approach for integrating LLMs into a multi-agent framework for autonomous driving control system. LLMs have the potential to provide common sense or reasoning ability that traditional systems lack, helping AVs better understand the context of complex real-world scenarios. Incorporating Vehicle-to-Vehicle communication using multi-agent, which will allow vehicles to exchange information with other vehicles. This capability is also essential for creating realistic and complex driving scenarios.

III. PRELIMINARY



Recommendation	Model Name	Benchmark Performance
LM Studio Recommended Models	Mistral 7B	74.6% (MMLU), 83.5% (GSM8K)
	OpenHermes 2.5	72.1% (MMLU), 79.2% (BIG-bench)
	DeepSeek 7B	76.3% (MMLU), 85.4% (GSM8K)
Hugging Face Open-Source Models	LLaMA 2 13B	75.8% (MMLU), 80.7% (BIG-bench)
	Mixtral 8x7B	78.2% (MMLU), 88.1% (GSM8K)
Top Commercial Services	Claude 3 Sonnet	82.3% (MMLU), 90.2% (BIG-bench)
	GPT-4 Turbo	85.6% (MMLU), 92.4% (GSM8K)
	Falcon 7B/40B	73.4% (MMLU), 78.9% (BIG-bench)

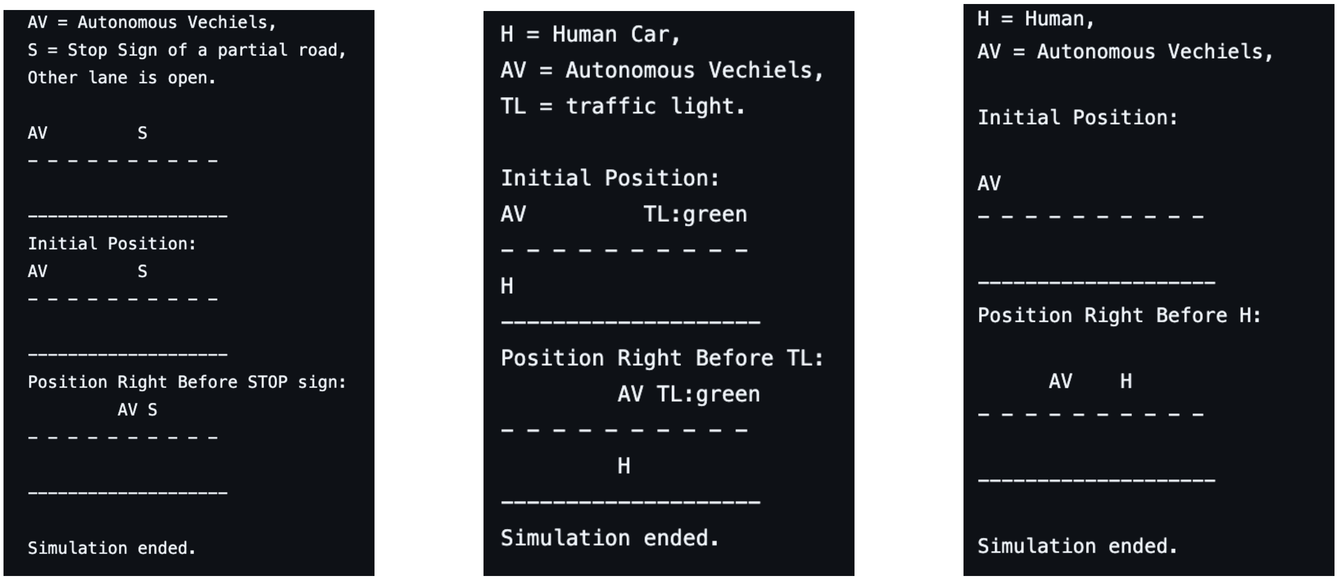
Fig. 3: Selected Models for the Experimentation

To investigate the use of LLMs as a decision agent in AVs, we have utilized open-source and local deployment platforms. Hugging Face[1] offers a wide range of open-source pre-trained models, while LM Studio[2] integrates LLM models from Hugging Face into a local system, even without powerful GPUs. Now, the question is what factors were considered in the selection of LLMs for this study? One key factor in

---

[1] https://huggingface.co/

[2] https://lmstudio.ai/



```
AV = Autonomous Vechiels,
S = Stop Sign of a partial road,
Other lane is open.

AV        S
- - - - - - - - - -


-------------------
Initial Position:
AV        S
- - - - - - - - - -


-------------------
Position Right Before STOP sign:
        AV S
- - - - - - - - - -


-------------------


Simulation ended.
```

(a) Stop sign on Partial road

```
H = Human Car,
AV = Autonomous Vechiels,
TL = traffic light.

Initial Position:
AV         TL:green
- - - - - - - - - -
H
--------------------
Position Right Before TL:
         AV TL:green
- - - - - - - - - -
         H
--------------------
Simulation ended.
```

(b) Traffic Light

```
H = Human,
AV = Autonomous Vechiels,

Initial Position:

AV
- - - - - - - - - -


--------------------
Position Right Before H:


      AV    H
- - - - - - - - - -


--------------------


Simulation ended.
```

(c) Human suddenly crossing road

Fig. 4: Illustration of real life traffic scenarios

our model selection was quantization, which optimizes model performance while reducing computational requirements. This research [9] indicates that 8-bit quantization enables the majority of LLMs to maintain a performance level comparable to their nonquantized equivalents, regardless of model size (e.g. 7B to 70B parameters). Moreover, LLMs that are quantized to 4 bits can also uphold similar performance to their non-quantized versions across most benchmarks. This approach achieves memory reduction of 50 to 75% while preserving precision in complex tasks such as reasoning, decision making, and domain-specific applications. The models chosen for this investigation were selected using a structured approach based on three key factors: popularity and performance in text-to-text generation, LM Studio recommendations for optimized accuracy, and comparative evaluations of the top commercial services. Specifically, 1) Some models were chosen based on the highest number of downloads from Hugging Face, ensuring widespread adoption and benchmark effectiveness. 2) Models suggested by LM Studio were included due to their strong performance and compatibility with local inference environments. 3) The best commercial services were selected based on their comparative performance with open-source counterparts, prioritizing accuracy, interpretability, and real-time inference. The selected models are shown in Fig. 3. However, not all Hugging Face models are fully compatible with the LM Studio runtime. As a result, some models were tested directly on the Hugging Face interface to avoid compatibility issues. Furthermore, while models such as OpenAI's o1, DeepSeek R1, and Llama 3.1 405B have demonstrated strong benchmark performance, they were not included in this study due to limited quantization support, lack of local deployment feasibility, and restricted open-source availability.

To test these selected models, we have prepared a set of simple text-based simulation scenarios using Python, as illustrated in Fig. 4. These scenarios are designed to cover situations that require logical reasoning or the application of common rules knowledge, such as recognizing a red traffic light and stopping accordingly. These scenarios will be used as input for various LLMs to assess how effectively they handle both types of challenges. The experiments will be conducted using the LLM selected in Fig.3.

IV. OUR APPROACH

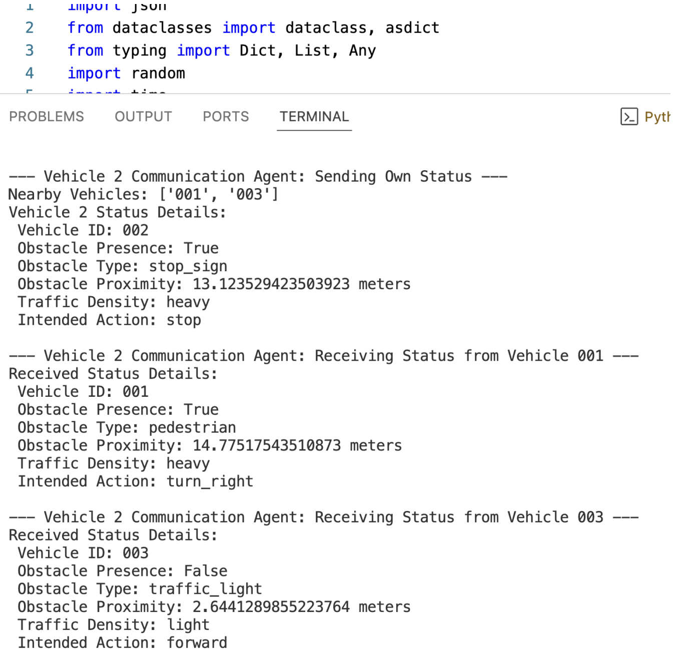
The main phase of this research focuses on using the shared data of the communication agent within our proposed multi-agent framework. The Communication Agent ensures that vehicles share critical information, including obstacle detection, traffic conditions, speed, and intended actions, in a standardized format within a defined radius, the expected format is demonstrated in Fig. 5. This structured data exchange is expected to significantly improve the accuracy of Decision Agent by providing a more comprehensive contextual understanding of the driving environment. For initial testing, we selected a basic traffic scenario involving an autonomous vehicle (AV) approaching a green traffic light, as illustrated in Fig. 4(b), and presented it to various language models to evaluate their decision-making behavior. The scenario was communicated through the prompt: 'What would the AV on the above do? Please just answer STOP or FORWARD.' The selected models included Claude[3] 3.7 Sonnet, GPT[4]-4 Turbo, Falcon[5] 3 7B and open-source alternatives such as OpenHermes-2.5-Mistral-7B and google/flan-t5-large. The responses of these models, sourced from Hugging Face, LM

---

[3] https://claude.ai/chats

[4] https://chat.openai.com/

[5] https://chat.falconllm.tii.ae/



```
import json
from dataclasses import dataclass, asdict
from typing import Dict, List, Any
import random
```

```
--- Vehicle 2 Communication Agent: Sending Own Status ---
Nearby Vehicles: ['001', '003']
Vehicle 2 Status Details:
 Vehicle ID: 002
 Obstacle Presence: True
 Obstacle Type: stop_sign
 Obstacle Proximity: 13.123529423503923 meters
 Traffic Density: heavy
 Intended Action: stop

--- Vehicle 2 Communication Agent: Receiving Status from Vehicle 001 ---
Received Status Details:
 Vehicle ID: 001
 Obstacle Presence: True
 Obstacle Type: pedestrian
 Obstacle Proximity: 14.77517543510873 meters
 Traffic Density: heavy
 Intended Action: turn_right

--- Vehicle 2 Communication Agent: Receiving Status from Vehicle 003 ---
Received Status Details:
 Vehicle ID: 003
 Obstacle Presence: False
 Obstacle Type: traffic_light
 Obstacle Proximity: 2.6441289855223764 meters
 Traffic Density: light
 Intended Action: forward
```

Fig. 5: An illustration of Communication Agent

Studio, and commercial providers, are shown in Fig. 11. GPT-4 Turbo and Falcon provided direct and expected responses ‘FORWARD’, which aligns with the logic of the situation. However, Claude deviated from the expected format by including an explanatory response, despite the prompt requesting a one-word answer. OpenHermes-2.5 responded with a clear ‘FORWARD’, demonstrating alignment with the input instructions. Surprisingly, the google/flan-t5-large output differed from all others by responding ‘STOP’, contradicting the intended logic of the scenario.

A major consideration was consistency. The subresearch question, ‘Does repeated querying of LLMs on the same scenario lead to variations in decision-making outcomes?’ will be one of such investigations of consistency. To test this, each scenario was submitted to the models multiple times for regenerating the outputs. This approach allowed us to observe whether the models produced stable and repeatable responses or if their decisions varied unpredictably. Consistency is especially critical in autonomous vehicle (AV) applications, where uncertain output can cause serious safety concerns.

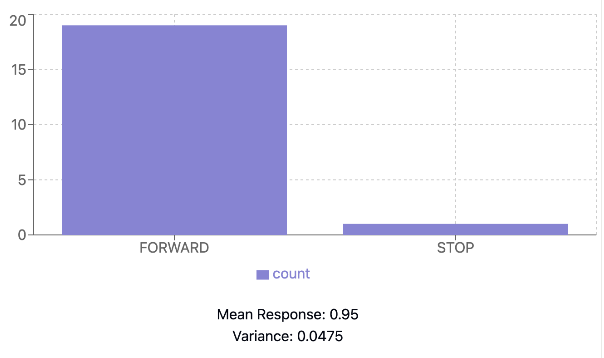
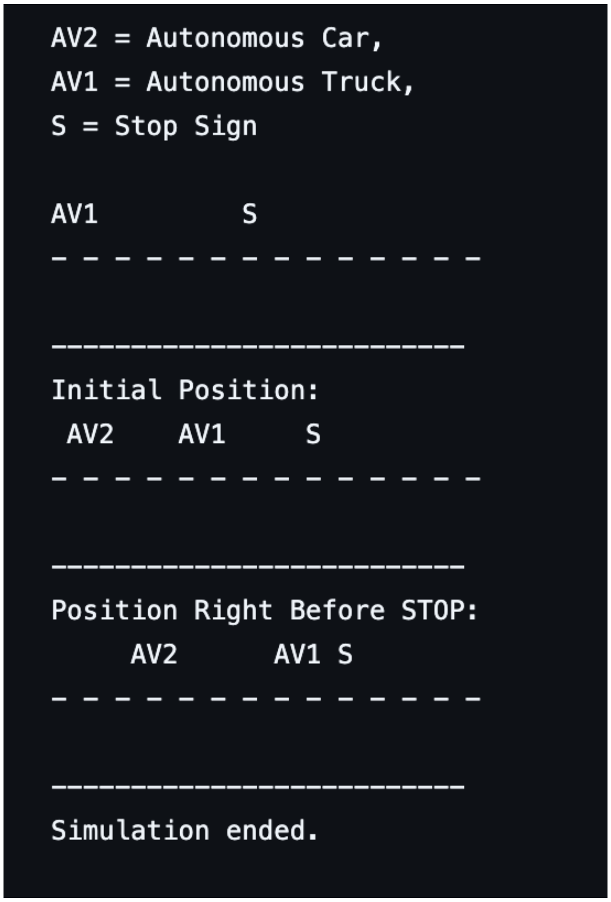


Fig. 6: LLM Model Consistency on the Tested Scenario

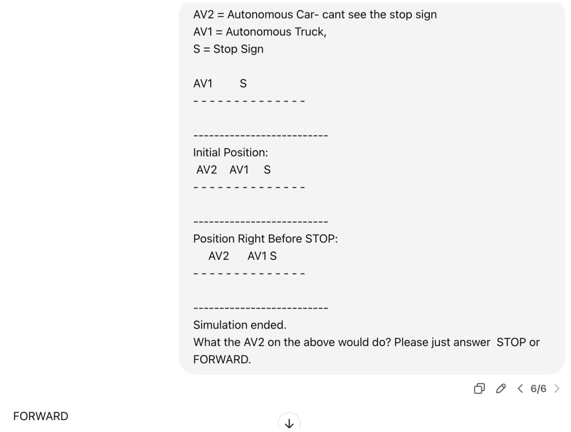
For giving an example of this analysis, we ran the prompt at least 20 times on OpenHermes-2.5-Mistral-7B, and the consistency results are presented in Fig. 6. Consistency was calculated by mapping ‘FORWARD’ = 1 and ‘STOP’ = 0. ‘FORWARD’ mean is closer to 1, it indicates that the output is having greater consistency with the expected response.



```
AV2 = Autonomous Car,
AV1 = Autonomous Truck,
S = Stop Sign

AV1         S
- - - - - - - - - - - - - -


--------------------------
Initial Position:
 AV2    AV1     S
- - - - - - - - - - - - - -


--------------------------
Position Right Before STOP:
     AV2      AV1 S
- - - - - - - - - - - - - -


--------------------------
Simulation ended.
```

Fig. 7: Extended Traffic Scenario with V2V Communication

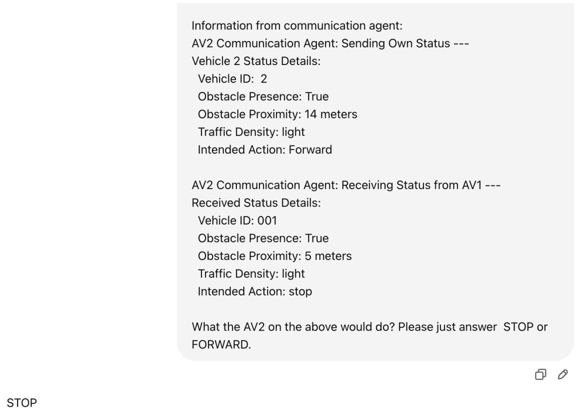
To investigate the impact of V2V communication on autonomous decision-making, we extended our simulation to a more complex traffic scenario. In this extended scenario, AV2 (an autonomous car) follows AV1 (an autonomous truck), with a stop sign ahead that AV2 cannot see because its line of sight is blocked by AV1, as shown in Fig. 7. To investigate how LLMs process and utilize this information, this scenario was presented to the LLM with the same prompt: ‘What would the AV on the above do? Please, just answer STOP or FORWARD’. The model responded with ‘FORWARD’, as shown in Fig. 8, for the case of using GPT-4 Turbo. This decision could potentially lead to an accident. We then examine whether the incorporation of V2V exchanged data improves the accuracy of the decisions and enables AVs to make more precise and context-aware decisions. By enabling



```
AV2 = Autonomous Car- cant see the stop sign
AV1 = Autonomous Truck,
S = Stop Sign

AV1       S
- - - - - - - - - - - - - - -


--------------------------
Initial Position:
 AV2   AV1    S
- - - - - - - - - - - - - - -


--------------------------
Position Right Before STOP:
    AV2     AV1 S
- - - - - - - - - - - - - - -


--------------------------
Simulation ended.
What the AV2 on the above would do? Please just answer  STOP or
FORWARD.
```

```
FORWARD
```

Fig. 8: LLM Model response without Communication agent's information (GPT-4 Turbo)



```
Information from communication agent:
AV2 Communication Agent: Sending Own Status ---
Vehicle 2 Status Details:
  Vehicle ID:  2
  Obstacle Presence: True
  Obstacle Proximity: 14 meters
  Traffic Density: light
  Intended Action: Forward

AV2 Communication Agent: Receiving Status from AV1 ---
Received Status Details:
  Vehicle ID: 001
  Obstacle Presence: True
  Obstacle Proximity: 5 meters
  Traffic Density: light
  Intended Action: stop

What the AV2 on the above would do? Please just answer  STOP or
FORWARD.
```

```
STOP
```

Fig. 9: LLM Model response with Communication agent's information (GPT-4 Turbo)

the Communication Agent, AV1 shares its intended action and information about the obstacle ahead with AV2. These data are structured and passed as additional input presented to the model, as shown in Fig. 9, for the case of using GPT-4 Turbo. With access to this contextual information, the model responds with 'STOP', even without direct visual confirmation of the traffic sign in the case where GPT-4 Turbo was used.

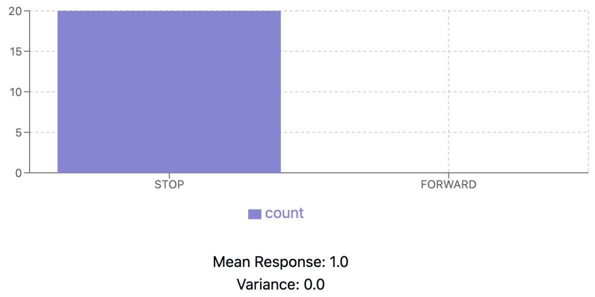


Fig. 10: LLM Model Consistency with V2V Communication support (GPT-4 Turbo)

To evaluate the consistency of the model decision, we ran the same prompt 20 times. The consistency was calculated by mapping 'FORWARD' = 1 and 'STOP' = 0. The resulting mean value of 1 for 'STOP' indicates that the output is having greater consistency with the expected response. The consistency of GPT-4 Turbo's responses with V2V support is shown in Fig. 10, which confirms that communication agent inputs help LLMs generate more accurate decisions in critical traffic environments.

V. CONCLUSION

We presented an approach for utilizing Large Language Models (LLMs) to enhance reasoning capabilities in uncertain traffic situations within a multi-agent framework for autonomous vehicles. We have investigated LLMs from Hugging Face, LM Studio, as well as major commercial services such as Claude, GPT-4 Turbo, and Falcon. Their responses to basic traffic scenarios were analyzed, revealing varying levels of consistency and accuracy. These results revealed both the promise and the current limitations of this approach, highlighting the need for improved consistency in the model responses. The proposed approach aims at the realization of adaptive autonomous systems capable of human-like reasoning in unpredictable situations. By integrating contextual data derived from V2V communication with LLM reasoning, we can have further investigations to bridge the critical gap in decision-making accuracy for autonomous vehicles.

REFERENCES

[1] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, *et al.*, "A survey on evaluation of large language models," *ACM transactions on intelligent systems and technology*, vol. 15, no. 3, pp. 1–45, 2024.

[2] A. Iftekhar and N. Fukuta, "Toward an efficient and resilient electric vehicle charging place allocation using multiagent approach," in *2024 16th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pp. 663–664, 2024.

[3] S. S. Duddela and N. Fukuta, "Toward an acceptable dynamic allocation considering fairness on autonomous car parking," in *2024 16th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI)*, pp. 286–291, 2024.

[4] J. Cho and J. Heo, "The more you know, the more you can trust: Drivers' understanding of the advanced driver assistance system," in *HCI in Mobility, Transport, and Automotive Systems. Automated Driving and In-Vehicle Experience Design: Second International Conference, MobiTAS 2020, Held as Part of the 22nd HCI International Conference, HCII 2020, Copenhagen, Denmark, July 19–24, 2020, Proceedings, Part I 22*, pp. 230–248, Springer, 2020.

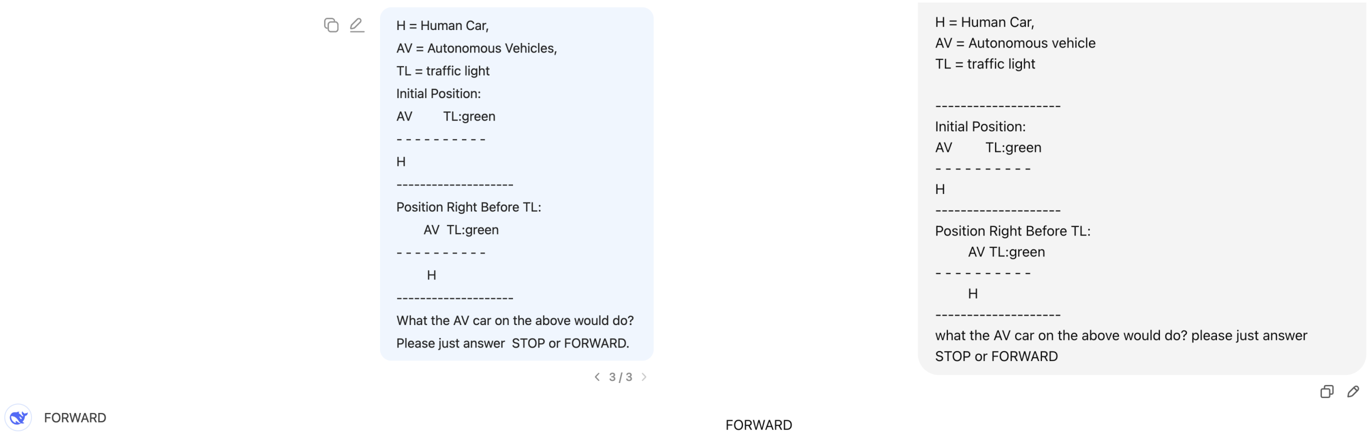
[5] M. Raza, "Autonomous vehicles: levels, technologies, impacts and concerns," *International Journal of Applied Engineering Research*, vol. 13, no. 16, pp. 12710–12714, 2018.

[6] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[7] D. Fu, X. Li, L. Wen, M. Dou, P. Cai, B. Shi, and Y. Qiao, "Drive like a human: Rethinking autonomous driving with large language models," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 910–919, 2024.

[8] T. Singh, E. van Hassel, A. Sheorey, and M. Alirezaei, "A systematic approach for creation of sotif's unknown unsafe scenarios: An optimization based method," tech. rep., SAE Technical Paper, 2024.

[9] G. Xiao, J. Lin, M. Seznec, H. Wu, J. Demouth, and S. Han, "Smoothquant: Accurate and efficient post-training quantization for large language models," in *International Conference on Machine Learning*, pp. 38087–38099, PMLR, 2023.



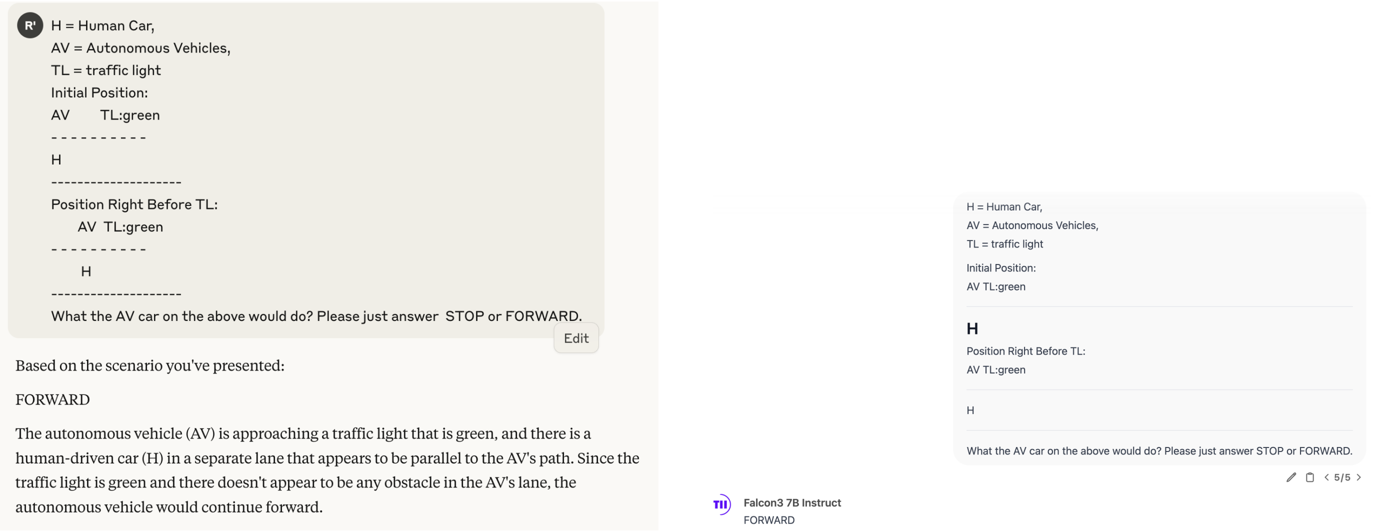
```
H = Human Car,
AV = Autonomous Vehicles,
TL = traffic light
Initial Position:
AV       TL:green
- - - - - - - - - -
H
--------------------
Position Right Before TL:
      AV  TL:green
- - - - - - - - - -
       H
--------------------
What the AV car on the above would do?
Please just answer  STOP or FORWARD.
```

FORWARD

```
H = Human Car,
AV = Autonomous vehicle
TL = traffic light

--------------------
Initial Position:
AV       TL:green
- - - - - - - - - -
H
--------------------
Position Right Before TL:
      AV TL:green
- - - - - - - - - -
      H
--------------------
what the AV car on the above would do? please just answer
STOP or FORWARD
```

FORWARD

(a) Output from DeepSeek-V3 (b) Output from GPT-4 turbo



```
H = Human Car,
AV = Autonomous Vehicles,
TL = traffic light
Initial Position:
AV       TL:green
- - - - - - - - - -
H
--------------------
Position Right Before TL:
      AV  TL:green
- - - - - - - - - -
      H
--------------------
What the AV car on the above would do? Please just answer  STOP or FORWARD.
```

Based on the scenario you've presented:

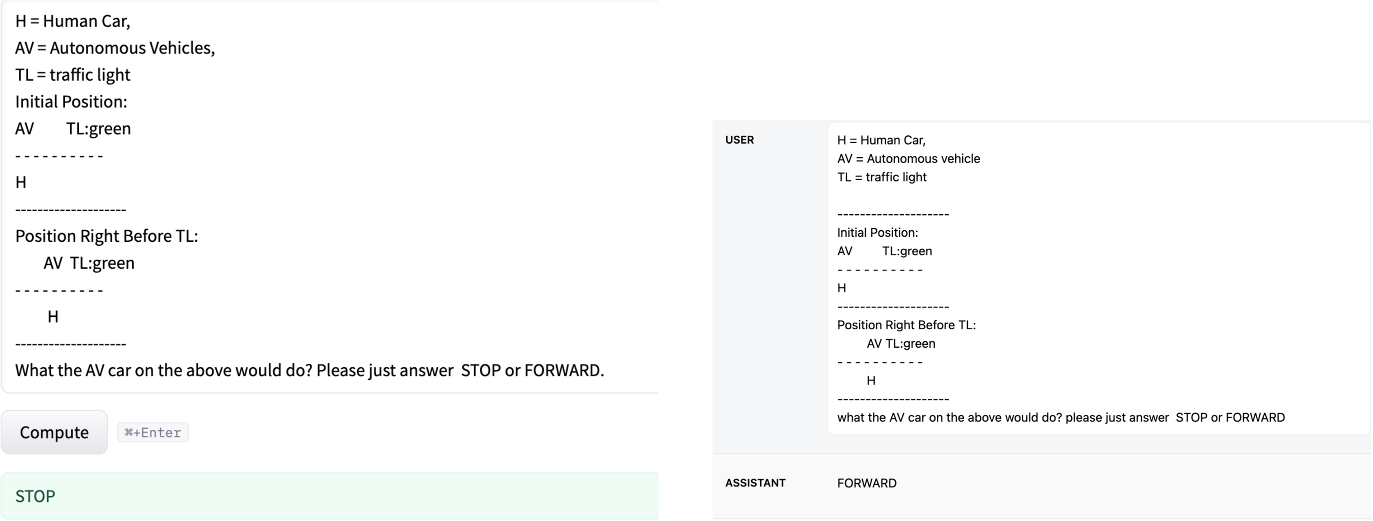
FORWARD

The autonomous vehicle (AV) is approaching a traffic light that is green, and there is a human-driven car (H) in a separate lane that appears to be parallel to the AV's path. Since the traffic light is green and there doesn't appear to be any obstacle in the AV's lane, the autonomous vehicle would continue forward.

```
H = Human Car,
AV = Autonomous Vehicles,
TL = traffic light
Initial Position:
AV TL:green

H
Position Right Before TL:
AV TL:green

H

What the AV car on the above would do? Please just answer STOP or FORWARD.
```

Falcon3 7B Instruct
FORWARD

(c) Output from Claude 3.7 Sonnet (d) Output from Falcon3 7B



```
H = Human Car,
AV = Autonomous Vehicles,
TL = traffic light
Initial Position:
AV       TL:green
- - - - - - - - - -
H
--------------------
Position Right Before TL:
      AV  TL:green
- - - - - - - - - -
      H
--------------------
What the AV car on the above would do? Please just answer  STOP or FORWARD.
```

STOP

USER

```
H = Human Car,
AV = Autonomous vehicle
TL = traffic light

--------------------
Initial Position:
AV       TL:green
- - - - - - - - - -
H
--------------------
Position Right Before TL:
      AV TL:green
- - - - - - - - - -
      H
--------------------
what the AV car on the above would do? please just answer  STOP or FORWARD
```

ASSISTANT FORWARD

(e) Output from google/flan-t5-large (f) Output from OpenHermes-2.5-Mistral-7B

Fig. 11: Responses from different LLMs

Acknowledgements

Alhamdulillah. In the name of Allah, the Most Gracious and the Most Merciful.

First of all, I would like to express my gratitude to Almighty God ALLAH SWT for the blessing, kindness and inspiration that guided me to complete the course.

I would like to express my sincere gratitude to my supervisor **Professor FUKUTA Naoki** for his motivation and enormous contribution of knowledge. His guidance and support were invaluable to the success of this work. This study is the result of guidance from my Supervisor. I want to acknowledge and give my sincerest gratitude and heartfelt appreciation for being my Supervisor and allowing me to be a part of Fukuta lab. Our professor consistently supervised me with excellent guidance on my research, which also aided me in conducting extensive studies and learning about bunches of new topics. The encouragement, advice, support, and most importantly, the belief in us, from the initial to the final phase, helped us a lot to understand the subject. Besides my supervisors, I would like to thank my parents for their blessing, my lab members, and my friend for their support. I am incredibly grateful to all the lab members of Fukuta Laboratory and my friends at Shizuoka University for always being supportive and friendly. They are very hardworking, and I believe their academic achievement will continue to increase. Thank you for comforting me with every difficulty and suggesting the proper way to deal with it. They helped me to adopt a new culture and daily life. I am also eternally grateful to all S-port staff who lovingly and carefully advised us.

Furthermore, I want to thank my respected advisory committee **....** and **...** for giving me very helpful comments and questions at the time of the plan presentation, the mid-term presentation, and the final presentation.

I am incredibly grateful to the faculty and administrators of the **Shizuoka University's Asia Bridge Program (ABP-SU)** for the golden opportunity to be a part of it.

Finally, words are insufficient to express gratitude to my cherished family members for supporting me and encouraging me.

Hamamatsu, Japan
July 14, 2025
Rafi Md Ashifujjman

Chapter 5

Simulation Source Code

5.1 Scenario 1: Traffic Light Recognition Simulation - Autonomous Vehicle Approaching Green Traffic Light

CODE LISTING 5.1: Scenario 1: Traffic Light Recognition Simulation - Autonomous Vehicle Approaching Green Traffic Light

```
# This code simulates a road with three lanes: top (AV), middle (divi
# The AV moves forward in the top lane, the human moves forward in the
# and the traffic light is in the middle-right.
# The code writes the road state to a file and moves the vehicles forw
# The simulation ends when the AV is close to the traffic light.

import time
class RoadCollisionSimulator:
    def __init__(self, road_length):
        # Initialize the road with 3 lanes: top (AV), middle (divider,
        self.road = [['␣' for _ in range(road_length)] for _ in range

        # Initial positions
        self.av_position = 0
        # AV starts at the beginning of the top lane
        self.human_position = 0
        # Human starts at the beginning of the bottom lane
        self.traffic_light_position = road_length // 2 + 2
        # Traffic light in the middle-right

        # Place vehicles and traffic light in initial positions
        self.road[0][self.av_position] = 'AV'
        # Place AV in the top lane
        self.road[0][self.traffic_light_position] = 'TL:green'
        # Place traffic light in top lane
        self.road[2][self.human_position] = 'H'
        # Place Human in the bottom lane

        # Lane divider setup (alternating '-' and ' ')
        self.road[1] = ['-' if i % 2 == 0 else '␣' for i in range(roa

        # Open a file to write the simulation output
        self.output_file = open("TrafficLightRecog.txt", "w")

    def write_road_to_file(self, description):
        # Write the description and road state to the file
        self.output_file.write(description + "\n")
        for lane in self.road:
            road_line = ''.join(lane)
            self.output_file.write(road_line + "\n")
            # Write each lane to the file
        self.output_file.write('-' * len(self.road[0]) + "\n")
        # Separator line
        self.output_file.flush()
        # Ensure data is written to the file

    def move_av(self):
        # Move AV forward in the top lane
        if self.av_position < self.traffic_light_position - 2:
            self.road[0][self.av_position] = '␣'
            # Remove AV from current position
            self.av_position += 1
            # Move AV forward
            self.road[0][self.av_position] = 'AV'
            # Place AV in the new position

    def move_human(self):
        # Move Human forward in the bottom lane
        if self.human_position < len(self.road[2]) - 1:
            self.road[2][self.human_position] = '␣'
            # Remove Human from current position
            self.human_position += 1
            # Move Human forward
            self.road[2][self.human_position] = 'H'
            # Place Human in the new position

    def simulate(self):
        # Step 1: Initial position
        self.write_road_to_file("AV␣=␣Autonomous␣Vehicle,\nH␣=␣Human,\
        self.write_road_to_file("Initial␣Position:")

        # Move vehicles until AV is close to traffic light
        while self.av_position < self.traffic_light_position - 2:
            self.move_av()
            self.move_human()
            time.sleep(0.3)

        # Step 2: Position right before traffic light
        self.write_road_to_file("Position␣Right␣Before␣TL:")

        # Step 3: Final movement
        self.move_av()
        self.move_human()

        # Step 3: Final position
        self.write_road_to_file("Final␣Position:")

        self.output_file.write("Simulation␣ended.\n")
        # Write simulation end message
        self.output_file.close()
        # Close the file after simulation ends

# Simulation parameters
road_length = 20

# Create an instance of the RoadCollisionSimulator
simulator = RoadCollisionSimulator(road_length)

# Run the simulation
simulator.simulate()
```

5.2 Scenario 2: Stop Sign Detection Simulation - Autonomous Vehicle Approaching Stop Sign on Partial Road

CODE LISTING 5.2: Scenario 2: Stop Sign Detection Simulation - Autonomous Vehicle Approaching Stop Sign on Partial Road

```
# This code simulates a road with two lanes: top (AV and Stop Sign),
# The AV moves forward in the top lane, the stop sign is in the middl
# and the divider is in the bottom lane.
# The code writes the road state to a file and moves the vehicles forw
# The simulation ends when the AV is close to the stop sign.

import time

class RoadCollisionSimulator:
    def __init__(self, road_length):
         # Initialize the road with 2 lanes: top (AV and Stop Sign), b
         self.road = [['␣' for _ in range(road_length)] for _ in range

         # Initial positions
         self.av_position = 0
         # AV starts at the beginning of the top lane
         self.stop_sign_position = road_length // 2
         # Stop sign in the middle of top lane

         # Place AV and stop sign in initial positions
         self.road[0][self.av_position] = 'AV'
         # Place AV in the top lane
         self.road[0][self.stop_sign_position] = 'S'
         # Place stop sign in top lane

         # Lane divider setup (alternating '-' and ' ') - represents t
         self.road[1] = ['-' if i % 2 == 0 else '␣' for i in range(roa

         # Open a file to write the simulation output
         self.output_file = open("StopSign.txt", "w")

    def write_road_to_file(self, description):
         # Write the description and road state to the file
         self.output_file.write(description + "\n")
         for lane in self.road:
             road_line = ''.join(lane)
             self.output_file.write(road_line + "\n")
             # Write each lane to the file
         self.output_file.write('-' * len(self.road[0]) + "\n")
         # Separator line
         self.output_file.flush()
         # Ensure data is written to the file

    def move_av(self):
         # Move AV forward in the top lane towards stop sign
         if self.av_position < self.stop_sign_position - 1:
             self.road[0][self.av_position] = '␣'
             # Remove AV from current position
             self.av_position += 1
             # Move AV forward
             self.road[0][self.av_position] = 'AV'
             # Place AV in the new position

    def simulate(self):
         # Header information
         self.write_road_to_file("AV␣=␣Autonomous␣Vehicles,\nS␣=␣Stop␣

         # Step 1: Initial position
         self.write_road_to_file("Initial␣Position:")

         # Move AV until it's right before the stop sign
         while self.av_position < self.stop_sign_position - 1:
             self.move_av()
             time.sleep(0.3)

         # Step 2: Position right before stop sign
         self.write_road_to_file("Position␣Right␣Before␣STOP␣sign:")

         self.output_file.write("Simulation␣ended.\n")
         # Write simulation end message
         self.output_file.close()
         # Close the file after simulation ends

# Simulation parameters
road_length = 15

# Create an instance of the RoadCollisionSimulator
simulator = RoadCollisionSimulator(road_length)

# Run the simulation
simulator.simulate()
```

5.3 Scenario 3: Sudden Human Crossing Detection Simulation - Autonomous Vehicle Encountering Unexpected Pedestrian

CODE LISTING 5.3: Scenario 3: Sudden Human Crossing Detection Simulation - Autonomous Vehicle Encountering Unexpected Pedestrian

```
# This code simulates a road with two lanes: top (AV and Human), botto
# The AV moves forward in the top lane, the human appears suddenly in
# and the divider is in the bottom lane.
# The code writes the road state to a file and moves the vehicles forw
# The simulation ends when the AV is close to where the human will ap

import time

class RoadCollisionSimulator:
    def __init__(self, road_length):
        # Initialize the road with 2 lanes: top (AV and Human), botto
        self.road = [['␣' for _ in range(road_length)] for _ in range

        # Initial positions
        self.av_position = 0
        # AV starts at the beginning of the top lane
        self.human_position = road_length // 2 + 1
        # Human will appear later in the middle-right

        # Place AV in initial position
        self.road[0][self.av_position] = 'AV'
        # Place AV in the top lane

        # Lane divider setup (alternating '-' and ' ')
        self.road[1] = ['-' if i % 2 == 0 else '␣' for i in range(roa

        # Open a file to write the simulation output
        self.output_file = open("Suddenhumancrossing.txt", "w")

    def write_road_to_file(self, description):
        # Write the description and road state to the file
        self.output_file.write(description + "\n")
        for lane in self.road:
            road_line = ''.join(lane)
            self.output_file.write(road_line + "\n")
            # Write each lane to the file
        self.output_file.write('-' * len(self.road[0]) + "\n")
        # Separator line
        self.output_file.flush()
        # Ensure data is written to the file
    def move_av(self):
        # Move AV forward in the top lane
        if self.av_position < len(self.road[0]) - 1:
            self.road[0][self.av_position] = '␣'
            # Remove AV from current position
            self.av_position += 1
            # Move AV forward
            if self.road[0][self.av_position] != 'H':
                # Don't overwrite Human
                self.road[0][self.av_position] = 'AV'
                # Place AV in the new position

    def add_human(self):
        # Human suddenly appears in the road
        self.road[0][self.human_position] = 'H'

    def simulate(self):
        # Header information
        self.write_road_to_file("H␣=␣Human,\nAV␣=␣Autonomous␣Vehicles,

        # Step 1: Initial position (only AV visible)
        self.write_road_to_file("Initial␣Position:")

        # Move AV until it's close to where human will appear
        while self.av_position < self.human_position - 2:
            self.move_av()
            time.sleep(0.3)

        # Human suddenly crosses - add human to the scene
        self.add_human()

        # Step 2: Position right before Human (human has suddenly app
        self.write_road_to_file("Position␣Right␣Before␣H:")

        self.output_file.write("Simulation␣ended.\n")
        # Write simulation end message
        self.output_file.close()
        # Close the file after simulation ends

# Simulation parameters
road_length = 15

# Create an instance of the RoadCollisionSimulator
simulator = RoadCollisionSimulator(road_length)
```

```
# Run the simulation
simulator.simulate()
```

5.4 Scenario 4: Multi-Vehicle Collision Simulation - Autonomous Truck and Car Approaching Stop Sign with Possible Road Collision Simulation

CODE LISTING 5.4: Scenario 4: Multi-Vehicle Collision Simulation - Autonomous Truck and Car Approaching Stop Sign with Possible Road Collision Simulation

```
# This code simulates a road with two lanes: top (AVs), bottom (divide
# The AV1 truck moves forward in the top lane, the AV2 car moves forwa
# and the stop sign is in the middle of the top lane.
# The code writes the road state to a file and moves the vehicles forw
# The simulation ends when the AV1 truck hits the stop sign.

import time

class RoadCollisionSimulator:
    def __init__(self, road_length):
        # Initialize the road with 2 lanes: top (AVs), bottom (divider
        self.road = [['␣' for _ in range(road_length)] for _ in range(
        self.road[0][road_length // 2] = 'S'
        # Stop sign in the top lane (initially at the middle)

        # Position AV1 (truck) closer to stop sign, AV2 (car) further
        self.av1_position = road_length // 2 - 2
        # AV1 truck starts 2 positions before stop sign
        self.av2_position = 0
        # AV2 car starts at the beginning of the top lane

        self.road[0][self.av1_position] = 'AV1'
        # Place AV1 truck in the top lane
        self.road[0][self.av2_position] = 'AV2'
        # Place AV2 car in the top lane

        self.collision_occurred = False
        # Track if a collision occurred

        # Lane divider setup (alternating '-' and ' ')
        self.road[1] = ['-' if i % 2 == 0 else '␣' for i in range(roa

        # Open a file to write the simulation output
        self.output_file = open("ExtendedScenario.txt", "w")

    def write_road_to_file(self, description):
        # Write the description and road state to the file
        self.output_file.write(description + "\n")
        for lane in self.road:
            road_line = ''.join(lane)
            self.output_file.write(road_line + "\n")
            # Write each lane to the file
        self.output_file.write('-' * len(self.road[0]) + "\n")
        # Separator line
        self.output_file.flush()
        # Ensure data is written to the file

    def detect_collision(self):
        # Check if AV1 hits the stop sign in the top lane
        if self.road[0][self.av1_position] == 'S':
            # Collision detected, mark with X
            self.road[0][self.av1_position] = 'X'
            # Mark collision with 'X'
            self.collision_occurred = True
            return True
        return False

    def move_av2_car(self):
        # Move AV2 car forward in the top lane (only if no collision
        if not self.collision_occurred and self.av2_position < self.a
            self.road[0][self.av2_position] = '␣'
            # Remove AV2 from current position
            self.av2_position += 1
            # Move AV2 forward
            if self.av2_position < len(self.road[0]):
                self.road[0][self.av2_position] = 'AV2'
                # Place AV2 in the new position

    def move_av1_truck(self):
        # Move AV1 truck forward in the top lane
        if not self.collision_occurred:
            self.road[0][self.av1_position] = '␣'
            # Remove AV1 from current position
            self.av1_position += 1
            # Move AV1 forward
            if self.av1_position < len(self.road[0]):
                if self.road[0][self.av1_position] == 'S':
                    # Do not update AV1 to 'X' here; collision detecti
                    pass
                else:
                    self.road[0][self.av1_position] = 'AV1'
                    # Place AV1 in the new position

    def simulate(self):
        # Step 1: Initial position
        self.write_road_to_file("AV2␣=␣Autonomous␣Car,\nAV1␣=␣Autonom
        self.write_road_to_file("Initial␣Position:")

        # Step 2: Move to position right before STOP
        # Move vehicles until AV1 is one step away from stop sign
        stop_sign_position = len(self.road[0]) // 2
        while self.av1_position < stop_sign_position - 1:
            self.move_av2_car()
            self.move_av1_truck()
            time.sleep(0.5)
            # Simulate movement delay

        # Write the "Position Right Before STOP" state
        self.write_road_to_file("Position␣Right␣Before␣STOP:")

        # Step 3: Final move - AV1 hits the stop sign
        self.move_av2_car()
        self.move_av1_truck()

        # Detect collision and mark with X
        self.detect_collision()

        # Write final position
        self.write_road_to_file("Final␣Position:")

        self.output_file.write("Simulation␣ended.\n")
        # Write simulation end message
        self.output_file.close()
        # Close the file after simulation ends

# Simulation parameters
road_length = 20
```

```
# Create an instance of the RoadCollisionSimulator
simulator = RoadCollisionSimulator(road_length)

# Run the simulation
simulator.simulate()
```

5.5 Scenario 5: Multi-Agent Coordination Simulation - Three Autonomous Vehicles Coordinating Movement Around Human Obstacle

CODE LISTING 5.5: Scenario 5: Multi-Agent Coordination Simulation - Three Autonomous Vehicles Coordinating Movement Around Human Obstacle

```
# This code simulates a road with three lanes: top (AV1, H), middle (
# The AV1 moves forward in the top lane, the human moves forward in th
# and the AV2 moves forward in the bottom lane.
# The code writes the road state to a file and moves the vehicles forw
# The simulation ends when the AV1 is close to the human.

import time

class RoadCollisionSimulator:
    def __init__(self, road_length):
        # Initialize the road with 3 lanes: top (AV1, H), middle (div
        self.road = [['␣' for _ in range(road_length)] for _ in range

        # Initial positions
        self.av1_position = 0
        # AV1 starts at the beginning of the top lane
        self.human_position = road_length // 2 + 2
        # Human starts in the middle-right of top lane
        self.av2_position = 0
        # AV2 starts at the beginning of the bottom lane
        self.av3_position = 0
        # AV3 will appear later in the top lane

        # Place vehicles in initial positions
        self.road[0][self.av1_position] = 'AV1'
        # Place AV1 in the top lane
        self.road[0][self.human_position] = 'H'
        # Place Human in the top lane
        self.road[2][self.av2_position] = 'AV2'
        # Place AV2 in the bottom lane

        # Lane divider setup (alternating '-' and ' ')
        self.road[1] = ['-' if i % 2 == 0 else '␣' for i in range(road

        # Open a file to write the simulation output
        self.output_file = open("NextStep.txt", "w")

    def write_road_to_file(self, description):
        # Write the description and road state to the file
        self.output_file.write(description + "\n")
        for lane in self.road:
            road_line = ''.join(lane)
            self.output_file.write(road_line + "\n")
            # Write each lane to the file
        self.output_file.write('-' * len(self.road[0]) + "\n")
        # Separator line
        self.output_file.flush()
        # Ensure data is written to the file

    def move_av1(self):
        # Move AV1 forward in the top lane
        if self.av1_position < len(self.road[0]) - 1:
            self.road[0][self.av1_position] = '␣'
            # Remove AV1 from current position
            self.av1_position += 1
            # Move AV1 forward
            if self.road[0][self.av1_position] != 'H':
                # Don't overwrite Human
                self.road[0][self.av1_position] = 'AV1'
                # Place AV1 in the new position

    def move_av2(self):
        # Move AV2 forward in the bottom lane
        if self.av2_position < len(self.road[2]) - 1:
            self.road[2][self.av2_position] = '␣'
            # Remove AV2 from current position
            self.av2_position += 1
            # Move AV2 forward
            self.road[2][self.av2_position] = 'AV2'
            # Place AV2 in the new position

    def add_av3(self):
        # Add AV3 to the top lane at the beginning
        self.av3_position = 0
        self.road[0][self.av3_position] = 'AV3'

    def simulate(self):
        # Step 1: Initial position
        self.write_road_to_file("H␣=␣Human,\nAV1,AV2,AV3␣=␣Autonomous␣
        self.write_road_to_file("Initial␣Position:")

        # Move vehicles until AV1 is close to Human
        while self.av1_position < self.human_position - 2:
            self.move_av1()
            self.move_av2()
            time.sleep(0.3)

        # Step 2: Position right before Human
        self.write_road_to_file("Position␣Right␣Before␣Human:")

        # Step 3: Final movements and add AV3
        self.move_av1()
        self.move_av2()
        self.add_av3()
        # AV3 appears in the final position

        # Step 3: Final position
        self.write_road_to_file("Final␣Position:")

        self.output_file.write("Simulation␣ended.\n")
        # Write simulation end message
        self.output_file.close()
        # Close the file after simulation ends

# Simulation parameters
road_length = 20

# Create an instance of the RoadCollisionSimulator
simulator = RoadCollisionSimulator(road_length)

# Run the simulation
simulator.simulate()
```

5.6 Scenario 6: Vehicle-to-Vehicle (V2V) Communication System - Multi-Agent Information Sharing and Status Broadcasting

CODE LISTING 5.6: Scenario 6: Vehicle-to-Vehicle (V2V) Communication System - Multi-Agent Information Sharing and Status Broadcasting

```
import json
from dataclasses import dataclass, asdict
from typing import Dict, List, Any
import random
import time
import math

@dataclass
class VehicleStatus:
    # Comprehensive vehicle status data structure
    vehicle_id: str
    timestamp: float

    # Obstacle Detection
    obstacle_presence: bool
    obstacle_type: str
    obstacle_proximity: float
    # 0-15 meters

    # Traffic Conditions
    traffic_density: str
    # light, moderate, heavy
    road_conditions: str
    # normal, wet, construction, etc.

    # Vehicle Dynamics
    current_speed: float
    direction: str
    # forward, backward, stationary

    # Intended Action
    intended_action: str
    # forward, stop, turn_left, turn_right

    # Position for distance calculation
```

```
        x_coordinate: float
        y_coordinate: float

    class V2VCommunicationSystem:
        def __init__(self, communication_radius: float = 50.0):
            self.communication_radius = communication_radius
            self.vehicles = {}

        def add_vehicle(self, vehicle_id: str, x: float, y: float):
            # Add a vehicle to the communication system
            self.vehicles[vehicle_id] = {
                'position': (x, y),
                'status': None
            }

        def calculate_distance(self, vehicle1_id: str, vehicle2_id: str) -
            # Calculate distance between two vehicles
            pos1 = self.vehicles[vehicle1_id]['position']
            pos2 = self.vehicles[vehicle2_id]['position']
            return math.sqrt((pos1[0] - pos2[0])**2 + (pos1[1] - pos2[1])*

        def get_nearby_vehicles(self, vehicle_id: str) -> List[str]:
            # Find nearby vehicles within communication radius
            nearby_vehicles = []
            for other_id in self.vehicles.keys():
                if other_id != vehicle_id:
                    distance = self.calculate_distance(vehicle_id, other_i
                    if distance <= self.communication_radius:
                        nearby_vehicles.append(other_id)
            return nearby_vehicles

    class Vehicle2CommunicationAgent:
        def __init__(self, v2v_system: V2VCommunicationSystem):
            self.vehicle_id = "002"
            # Vehicle 2 (Sender)
            self.v2v_system = v2v_system

            # Position Vehicle 2 at (10, 10)
            self.v2v_system.add_vehicle(self.vehicle_id, 10, 10)

            # Tracking received statuses
            self.received_statuses = {}
            self.nearby_vehicles = []
            # Open output file for writing
            self.output_file = open("v2v_communication_output.txt", "w")

        def write_to_file(self, message):
            # print(message)
            self.output_file.write(message + "\n")
            self.output_file.flush()

        def send_own_status(self):
            # Generate and send Vehicle 2's own status with specific deta
            self.write_to_file("\n---␣Vehicle␣2␣Communication␣Agent:␣Sendi

            # Scan for nearby vehicles
            self.nearby_vehicles = self.v2v_system.get_nearby_vehicles(se
            self.write_to_file(f"Nearby␣Vehicles:␣{self.nearby_vehicles}")

            # Create status with specific requirements
            vehicle_2_status = VehicleStatus(
                vehicle_id="002",
                timestamp=time.time(),

                # Obstacle Detection with stop sign
                obstacle_presence=True,
                obstacle_type="stop_sign",
                obstacle_proximity=13.123529423503923,

                # Traffic Conditions
                traffic_density="heavy",
                road_conditions=random.choice(['normal', 'wet', 'construct

                # Vehicle Dynamics
                current_speed=random.uniform(0, 60),
                direction='forward',

                # Intended Action
                intended_action='stop',

                # Position
                x_coordinate=10,
                y_coordinate=10
            )

            # Store status in V2V system
            self.v2v_system.vehicles[self.vehicle_id]['status'] = asdict(
            self.write_to_file("Vehicle␣2␣Status␣Details:")
            self.write_to_file(f"␣Vehicle␣ID:␣{vehicle_2_status.vehicle_id
            self.write_to_file(f"␣Obstacle␣Presence:␣{vehicle_2_status.ob
            self.write_to_file(f"␣Obstacle␣Type:␣{vehicle_2_status.obstacl
            self.write_to_file(f"␣Obstacle␣Proximity:␣{vehicle_2_status.o
            self.write_to_file(f"␣Traffic␣Density:␣{vehicle_2_status.traff
            self.write_to_file(f"␣Intended␣Action:␣{vehicle_2_status.inten

            return vehicle_2_status

        def receive_vehicle_status(self, sender_vehicle_id: str):
            # Receive status from another vehicle
            self.write_to_file(f"\n---␣Vehicle␣2␣Communication␣Agent:␣Rec

            # Simulate receiving status from the sender vehicle
            sender_status = self.v2v_system.vehicles[sender_vehicle_id]['s

            self.write_to_file("Received␣Status␣Details:")
            self.write_to_file(f"␣Vehicle␣ID:␣{sender_status['vehicle_id']
            self.write_to_file(f"␣Obstacle␣Presence:␣{sender_status['obsta
            self.write_to_file(f"␣Obstacle␣Type:␣{sender_status.get('obsta
            self.write_to_file(f"␣Obstacle␣Proximity:␣{sender_status['obs
            self.write_to_file(f"␣Traffic␣Density:␣{sender_status['traffic
            self.write_to_file(f"␣Intended␣Action:␣{sender_status['intende

            # Store received status
            self.received_statuses[sender_vehicle_id] = sender_status

        def close_file(self):
            # Close the output file
            self.output_file.close()

    def simulate_v2v_communication():
        # Simulate V2V communication from Vehicle 2's perspective
        # Create V2V communication system
        v2v_system = V2VCommunicationSystem(communication_radius=50.0)

        # Create other vehicles
        v2v_system.add_vehicle("001", 5, 5)
        # Vehicle 1 close to Vehicle 2
        v2v_system.add_vehicle("003", 20, 20)
        # Vehicle 3 at (20, 20)
        # Create Vehicle 2's Communication Agent
        vehicle_2_comm = Vehicle2CommunicationAgent(v2v_system)

        # Vehicle 2 sends its own status
        vehicle_2_status = vehicle_2_comm.send_own_status()

        # Simulate sending status to other vehicles
        # (In a real system, this would be a broadcast)
        v2v_system.vehicles["001"]['status'] = {
            'vehicle_id': "001",
            'timestamp': time.time(),
            'obstacle_presence': random.choice([True, False]),
            'obstacle_type': random.choice(['traffic_light', 'pedestrian'
            'obstacle_proximity': random.uniform(0, 15),
            'traffic_density': random.choice(['light', 'moderate', 'heavy
            'road_conditions': random.choice(['normal', 'wet', 'constructi
            'current_speed': random.uniform(0, 120),
            'direction': random.choice(['forward', 'stationary']),
            'intended_action': random.choice(['forward', 'stop', 'turn_le
            'x_coordinate': 5,
            'y_coordinate': 5
        }

        v2v_system.vehicles["003"]['status'] = {
            'vehicle_id': "003",
            'timestamp': time.time(),
            'obstacle_presence': random.choice([True, False]),
            'obstacle_type': random.choice(['traffic_light', 'pedestrian'
            'obstacle_proximity': random.uniform(0, 15),
            'traffic_density': random.choice(['light', 'moderate', 'heavy
            'road_conditions': random.choice(['normal', 'wet', 'constructi
            'current_speed': random.uniform(0, 120),
            'direction': random.choice(['forward', 'stationary']),
            'intended_action': random.choice(['forward', 'stop', 'turn_le
            'x_coordinate': 20,
            'y_coordinate': 20
        }

        # Vehicle 2 receives statuses from Vehicle 1 and Vehicle 3
        vehicle_2_comm.receive_vehicle_status("001")
        vehicle_2_comm.receive_vehicle_status("003")

        # Write simulation completion message
        vehicle_2_comm.write_to_file("\n–V2V␣Communication␣Simulation␣Com
        # Close the output file
        vehicle_2_comm.close_file()

    # Run the simulation
    if __name__ == "__main__":
        simulate_v2v_communication()
```

5.7 Scenario 7: LLM Decision Consistency Analysis - Statistical Evaluation of Language Model Decision Patterns in Autonomous Driving Scenarios

CODE LISTING 5.7: Scenario 7: LLM Decision Consistency Analysis - Statistical Evaluation of Language Model Decision Patterns in Autonomous Driving Scenarios

```
import numpy as np
import matplotlib.pyplot as plt

# Response data (1 = FORWARD, 0 = STOP)
responses = [1] * 20
# All 20 responses were FORWARD
# Calculate statistics
mean_response = np.mean(responses)
variance_response = np.var(responses)
std_dev = np.std(responses)
# Create visualization
plt.figure(figsize=(8, 5))

# Bar plot for mean
plt.bar(['Mean'], [mean_response], color='skyblue', label=f'Mean:␣{me

# Add error bar showing standard deviation
plt.errorbar(['Mean'], [mean_response], yerr=[std_dev],
             fmt='none', ecolor='red', capsize=10, label=f'Std␣Dev:␣{st

# Customize plot
plt.title('LLM␣Decision␣Tendency␣(FORWARD=1,␣STOP=0)', pad=20)
plt.ylim(0, 1.2)
plt.ylabel('Response␣Value')
plt.legend()
plt.grid(axis='y', alpha=0.3)
```

```
    # Annotate important values
    plt.text(0, mean_response+0.05, f'Variance:␣{variance_response:.2f}',
            ha='center', va='bottom', fontsize=10)

plt.show()
```