



CHALMERS
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

Integrating a Neuro-Symbolic Planner for Safe Driving

Master's thesis in Computer science and engineering

Saghar Alvandi
Jennifer Nilsson

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

MASTER'S THESIS 2026

Integrating a Neuro-Symbolic Planner for Safe Driving

Saghar Alvandi
Jennifer Nilsson



UNIVERSITY OF
GOTHENBURG



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2026

Integrating a Neuro-Symbolic Planner for Safe Driving
Saghar Alvandi, Jennifer Nilsson

© Saghar Alvandi, Jennifer Nilsson, 2026.

Supervisor: Hazem Torfah & Antonina Skurka, Department of Computer Science
and Engineering

Advisor: Anders Björklund & Reza Javeheri, Volvo Trucks Technology & Industrial
Division

Examiner: Moa Johansson, Department of Computer Science and Engineering

Master's Thesis 2026

Department of Computer Science and Engineering

Chalmers University of Technology and University of Gothenburg

SE-412 96 Gothenburg

Telephone +46 31 772 1000

Typeset in L^AT_EX
Gothenburg, Sweden 2026

Integrating a Neuro-Symbolic Planner for Safe Driving
Saghar Alvandi, Jennifer Nilsson
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Ensuring the safety and reliability of autonomous driving systems remains an ongoing challenge. The complexity of real-world traffic scenarios highlights the importance of autonomous driving systems being able to reason about their surrounding environments and make safe decisions. This thesis proposes a modular neuro-symbolic framework that decouples trajectory prediction from motion planning. Rather than relying solely on an end-to-end deep-learning-based planner, the proposed approach uses a fine-tuned large language model to predict the future trajectories of surrounding actors. These predictions are incorporated as dynamic obstacles into a symbolic A* planner, which generates a goal-oriented trajectory for the ego vehicle subject to safety constraints. The proposed framework is evaluated on data generated from driving scenarios developed in Scenic and executed in the CARLA simulator, with its performance compared against an end-to-end deep-learning-based planner using the same set of 144 scenarios. The results demonstrate that this modular neuro-symbolic approach achieves a higher rate of safe, goal-reaching trajectories, with a success rate of 59.0% compared with 27.1% for the baseline planner. However, the neuro-symbolic planner also exhibits a higher collision rate of 32.6%, compared with 10.4% for the baseline planner. This difference is partly due to the baseline frequently producing safe but non-goal-reaching trajectories, with 57 scenarios falling into this category because the ego vehicle remains at least 100 meters away from the other actors throughout the scenario. These findings highlight a trade-off between collision avoidance and goal achievement, indicating the need for further evaluation to draw clearer conclusions regarding the safety of the approach.

Keywords: autonomous vehicles; large language models (LLMs); symbolic planning; trajectory prediction

Acknowledgements

This Master's thesis was carried out during the spring 2026 at Volvo Trucks Technology & Industrial Division in Gothenburg within the Services team. We would like to express our sincere gratitude to the individuals and groups whose support and contributions were invaluable to the completion of this work.

First and foremost, we would like to express our sincere appreciation to Hazem Torfah and Antonina Skurka, our supervisor at Chalmers University of Technology. His insightful guidance, constructive feedback, and continuous supervision were invaluable throughout the research and writing process.

Our heartfelt thanks go to Anders Björklund and Reza Javaheri, our supervisors at Volvo Group, for their invaluable project direction, expertise, and unwavering support. Their encouragement and guidance were a constant source of motivation throughout this thesis.

We are also thankful to Malin Larking, our responsible line manager at Volvo Group, for her efficient handling of administrative matters, her continuous support, and her role in the planning stages of this thesis.

Finally, we would like to express our deepest gratitude to our families and friends for their unwavering encouragement, patience, and support, not only throughout this thesis but throughout our entire academic journey. Their belief in us and constant motivation made this achievement possible.

Saghar Alvandi, Jennifer Nilsson, Gothenburg, 2026-10-20

Contents

List of Figures	xi
List of Tables	xiii
1 Introduction	1
1.1 Background	1
1.2 Purpose	2
1.2.1 Research questions	2
1.3 Project Scope	2
2 Related Work	4
2.1 LLMs for Decision Making in Autonomous Driving	4
2.1.1 SafeDrive	4
2.1.2 Limitations of Current LLM-based Approaches	5
2.2 Symbolic Planning for Decision-Making	6
2.2.1 Symbolic Planning in Autonomous Driving	6
2.3 Neuro-symbolic approaches	7
3 Theoretical Background	9
3.1 Large Language models	9
3.1.1 Terminology	9
3.2 Supervised Fine-Tuning (SFT)	9
3.2.1 Full Fine-tuning	10
3.2.2 Parameter-Efficient Fine-Tuning (PEFT)	10
3.3 Symbolic Planner	11
3.3.1 Search-Based Path Planning	11
3.3.2 A* Search Algorithm	11
4 Methods	14
4.1 System Overview	14
4.2 Dataset Generation	16
4.2.1 Scenario Creation	16
4.2.2 Data Collection	17
4.3 A Neuro-Symbolic Planner	17
4.3.1 An LLM for Predicting the Trajectories of Surrounding Actors	18
4.3.1.1 Model Selection and Preparation	18

4.3.1.2	Data Preprocessing	19
4.3.1.3	Fine-Tuning the LLM	20
4.3.1.4	Evaluation Metrics	21
4.3.2	A* Planner	22
4.3.2.1	Environment Representation	22
4.3.2.2	Road Representation	23
4.3.2.3	Trajectory Prediction Representation	24
4.3.2.4	Motion Planning with A*	24
4.3.2.5	Safety Verification	26
4.3.2.6	Cost Function and Heuristic	28
4.4	An End-to-End Neural Planner	29
4.4.1	Motivation and Design	29
4.4.2	Dataset Generation	30
4.4.3	Dataset Format	30
4.4.3.1	Input format	30
4.4.3.2	Output Format	31
4.4.4	Fine-Tuning of the End-to-End Planner	31
4.5	Testing of the Neuro-Symbolic and End-to-End Neural Planners	32
5	Results	33
5.1	LLM's Accuracy	33
5.2	End-to-End Planner vs Neuro-Symbolic Planner	35
5.3	End-to-End Planner	35
5.4	Neuro-Symbolic Planner	37
6	Conclusion	41
	Bibliography	43

List of Figures

3.1	Comparison of different fine-tuning methods and their memory requirements. QLoRA reduces memory consumption compared to LoRA by quantizing the pre-trained model weights to 4-bit precision and using paged optimizers to handle memory spikes. Source: Adapted from [15].	10
3.2	Illustration of the A* search principle. Candidate nodes are evaluated using the total estimated cost $f(n) = g(n) + h(n)$, where the node with the lowest evaluation value is selected for further exploration. . .	12
4.1	Overview of the proposed pipeline, including scenario generation, dataset preparation, QLoRA fine-tuning, and the evaluation of the neuro-symbolic and end-to-end planning approaches based on safety and goal completion.	15
4.2	Overview of the proposed neuro-symbolic planner.	18
4.3	Example of local ego-centric occupancy grid.	22
5.1	ADE across five prediction steps over a 2.5-second prediction horizon.	33
5.2	ADE across the five prediction steps, showing the mean, median, interquartile range, and high-error outliers.	34

List of Tables

4.1	QLoRA fine-tuning configuration for the LLM.	21
4.2	Parameters used for coordinate transformation	23
4.3	Occupancy grid cell states	23
4.4	Grid displacement for each action.	26
5.1	Result of the neuro-symbolic planner in comparison to the end-to-end (baseline) planner over 144 simulations.	35
5.2	End-to-end planner result across different scenario types, showing the number and its percentage of evaluated scenarios that safely reached the goal, remained safe but did not reach the goal, or resulted in a collision.	36
5.3	The end-to-end planner’s distribution of minimum distances between the ego vehicle trajectory and the ground truth positions of the surrounding actors for safe scenarios that did not reach the goal position.	37
5.4	Neuro-symbolic planner result across different scenario types, showing the number and its percentage of evaluated scenarios that safely reached the goal, remained safe but did not reach the goal, or resulted in a collision.	38
5.5	The neuro-symbolic planner’s distribution of minimum distances between the ego vehicle trajectory and the ground truth positions of the surrounding actors for safe scenarios that did not reach the goal position.	39

1

Introduction

1.1 Background

Autonomous driving technology continues to advance with each passing year as new approaches are emerging. Most autonomous vehicles (AVs) can already operate safely in common traffic scenarios [1]. Though despite these advances, challenges remain in ensuring the safety and reliability of their deployment [2].

Current deep learning (DL) models employed in AV systems perform well at detecting and localizing objects within the environment [3]. However, accurately perceiving surrounding objects does not necessarily mean having a sufficient understanding of complex traffic environments, where interpretation of the environment and anticipating the behavior of road users are essential. In particular, conventional perception models typically operate on low-level sensor features, which can limit their ability to perform semantic reasoning and contextual interpretation of complex driving scenarios [4]. Recently, research has explored the use of large language models (LLMs) for trajectory prediction in autonomous driving systems. Due to their semantic reasoning capabilities and ability to model sequential dependencies, LLMs show promise for predicting the future trajectories of interacting road users [5]. Furthermore, fine-tuning a pretrained LLM allows the model to be adapted to trajectory prediction tasks, providing a potential way to leverage these capabilities to model the future movements of surrounding actors.

Simultaneously, symbolic planning frameworks offer complementary strengths. They rely on structured world models and explicit representations of the environment to support predictable behavior [6]. On the other hand, their performance can degrade when the conditions underlying these models are broken [6]. By enforcing explicit safety constraints, such as collision avoidance, lane boundaries, speed limits, and time-to-collision thresholds, symbolic planners are well-suited for decision-making in safety-critical systems such as AVs.

However, symbolic planners require clean, structured, and semantically meaningful inputs to operate effectively. Extracting such inputs from raw perception data is difficult and labor-intensive, limiting their applicability to complex scenarios where the required preprocessing is expensive to obtain. An alternative approach that avoids this input bottleneck is end-to-end deep learning, in which neural networks directly map raw sensor data to planning actions without intermediate symbolic

representations [7]. These methods can adapt to different scenarios and learn from data without requiring manually engineered rules. However, end-to-end methods are difficult to interpret, do not inherently guarantee safety, and may perform poorly in unexpected situations, limiting their suitability for real-world autonomous driving.

Despite the complementary strengths of neural and symbolic approaches, they remain largely separate. There are few frameworks that tightly integrate the semantic reasoning capabilities of LLMs with the explicit constraints of symbolic planning. This motivates further investigation of approaches combining the flexibility of neural methods with the structured decision-making of symbolic planners.

1.2 Purpose

The purpose of this project is to develop and evaluate an integrated framework that combines the semantic reasoning capabilities of LLMs with the logical structure and explicit safety constraints of symbolic planning for autonomous driving.

To address this challenge, this project develops and evaluates a pipeline that:

1. Uses a fine-tuned LLM to predict the future trajectories for surrounding objects.
2. Integrates these predictions as dynamic obstacles into a symbolic A* planner to generate a goal-oriented trajectory for the ego vehicle, while accounting for safety constraints.

1.2.1 Research questions

The primary research question of this thesis is:

RQ1: Does the proposed neuro-symbolic approach improve safety in comparison to a standard end-to-end deep-learning-based planner?

This question is addressed through a quantitative experimental evaluation comparing the performance of the proposed approach to an end-to-end baseline planner.

To further examine the trajectory prediction component, a secondary research question is considered:

RQ2: How accurately do the LLM-predicted trajectories match the ground-truth trajectories of the surrounding actors?

1.3 Project Scope

This thesis focuses on developing and evaluating a neuro-symbolic framework for autonomous driving in simulated intersection scenarios. The work involves developing an LLM-based trajectory prediction module, integrating its predictions into a symbolic A* planner, and comparing the resulting framework against an end-to-end deep-learning-based baseline planner. The scenarios are developed using the Scenic language and executed in the CARLA simulator. The evaluation is then performed

using quantitative metrics to assess trajectory prediction accuracy, goal-reaching performance, and collision rates based on the generated trajectories of the planners and corresponding ground-truth coordinates. Real-world testing, extensions to other driving scenarios, formal safety verification, and comparison with other state-of-the-art methods are outside the scope of this work.

2

Related Work

This chapter focuses on three key areas: LLM-based decision-making, symbolic planning methods, and recent neuro-symbolic approaches. The goal is to highlight both the strengths and limitations of these methods, and to identify the gap that motivates the proposed approach.

2.1 LLMs for Decision Making in Autonomous Driving

The development of LLMs has introduced new approaches to decision-making in autonomous vehicles [1][8]. Compared to traditional methods that rely heavily on data and can be hard to explain, LLMs can process natural-language descriptions and provide contextual interpretations. Despite their advantages, LLMs raise safety concerns due to hallucinations, overconfidence, and a limited understanding of physical vehicle dynamics. To address these issues, researchers have developed modular frameworks that combine LLM-based reasoning with risk quantification or formal verification.

2.1.1 SafeDrive

LLMs are becoming increasingly relevant in recent research on autonomous vehicles, especially for scenario interpretation [1]. One example of such an approach is SafeDrive, a risk-sensitive decision-making framework that utilizes LLMs. This framework addresses the challenges of ensuring safety in difficult environments by implementing a risk module, a memory module, an LLM-powered reasoning module, and a reflection module. The study reports that the incorporation of LLMs enables better scene/contextual interpretation and human-like reasoning, improving performance compared to traditional rule-based or learning-based planners in risk-sensitive environments [1].

In particular, LLMs enable a semantic understanding of complex driving scenes by interpreting structured and unstructured inputs. This allows them to reason about interactions between multiple agents in a human-like manner. This capability is especially useful in scenarios where explicit rule-based formulations become difficult due to combinatorial complexity. This is particularly relevant in urban driving scenarios,

where the number of possible interactions between agents increases significantly and cannot easily be captured by predefined rules [1].

However, research indicates that an LLM-based planner performs less effectively when used in isolation, with a standalone LLM agent acting as an end-to-end decision-maker without external safety constraints, than when integrated into a modular framework [1]. In the SafeDrive framework for evidence, a baseline GPT-4 agent achieved a safety rate of only 69.23% to 77.46%, which may be due to overconfidence and difficulties handling high-risk scenarios [1].

To address these issues, the SafeDrive framework couples the LLM with three specialized components, a risk module, a memory module and a reflection module. These modules provide capabilities that the LLM lacks on its own. The risk module offers a mathematical quantification of perceiving risk (QPR) to prevent risky maneuvers, the memory module ensures consistency by retrieving past experiences, and the reflection module enables self-correction by analyzing and refining erroneous actions [1].

Furthermore, while knowledge-driven frameworks, such as SafeDrive, enable high-level semantic reasoning, they often struggle to explicitly model how the environment will evolve over time due to a limited understanding of physical laws. As a result, they may lack the ability to accurately predict the future trajectories of surrounding actors. To address this gap, frameworks like SANDRA integrate reachability analysis and set-based predictions to mathematically model all possible future behaviors of other traffic participants. This ensures that the LLM’s decisions are not only logically sound but also provably safe within the vehicle’s physical state space [1].

2.1.2 Limitations of Current LLM-based Approaches

These approaches highlight the effectiveness that LLMs can provide for contextual interpretations and reasoning, but also their limitations when used alone [8][1]. Despite these advances, many existing approaches still rely on LLMs mainly to generate discrete actions based on the current state of the environment. Safety is often handled separately, either through external modules or through post-hoc verification. This separation creates a gap between high-level reasoning and low-level planning [8]. As a result, it becomes difficult to incorporate predictive information directly into the decision-making process.

In addition, current methods rarely make full use of LLMs for reasoning about how a scene may evolve over time [8]. For example, they often do not explicitly consider how surrounding agents might move. They also do not incorporate such predictions when selecting actions.

Together, these observations highlight a key limitation in current approaches. While LLMs provide strong high-level reasoning capabilities, they are not tightly integrated with the planning process [8]. In practice, prediction and planning are still treated as separate components, and LLM outputs are rarely used to directly guide the planning process. This creates a gap between reasoning and planning.

This suggests a need for tighter integration, where LLM outputs can be transformed into structured representations that directly influence the planning process. Such integration would enable decisions that are not only context-aware, but also better aligned with future developments in the environment [8].

2.2 Symbolic Planning for Decision-Making

In autonomous driving, symbolic planning is commonly used as part of the motion planning module to generate safe and efficient trajectories [4]. The methods represent the driving situation in a structured way, where the current state includes information such as the vehicle’s position, speed, and surrounding traffic. Based on this, the planner can consider different possible actions, such as changing lanes or adjusting speed, and evaluates how each action would affect the future situation. This allows the system to reason about different possible actions before making a decision.

2.2.1 Symbolic Planning in Autonomous Driving

An overview of these planning approaches is given in [4], where methods are divided into rule-based, search-based, and optimization-based strategies. In practice, these planners generate candidate trajectories for the vehicle by considering factors such as lane structure, surrounding vehicles, and safety constraints. For example, a search-based planner may explore different possible paths, such as maintaining the current lane or performing a lane change, and evaluate them based on cost functions that capture safety, efficiency, and comfort. Among these approaches, search-based planners such as A* are particularly relevant, as they explicitly explore candidate trajectories and evaluate them using structured cost functions. In this thesis, the A* algorithm is described in Section 3.3.2

The planner then selects the trajectory with the lowest cost that satisfies all constraints, such as collision avoidance and speed limits. This process allows the system to reason about multiple future outcomes before executing an action, making it well-suited for structured environments like highways, where traffic rules and constraints are clearly defined.

At the same time, these approaches also have some clear limitations. Since they rely on predefined states and manually designed constraints, they can struggle in complex and constantly changing environments. In real-world driving, it is not possible to anticipate every possible situation in advance, which means that important interactions or edge cases may not be fully captured. In particular, these planners are typically provided only with current state information, not predictive models about future agent behavior.

Another limitation lies in how the environment is represented. Symbolic planners typically operate on structured inputs generated by perception modules, such as distances, speeds, and lane positions. Although this type of representation is effective for enforcing rules and safety constraints, its predefined structure can limit the ability to capture higher-level aspects of the situation, causing a human-defined information

bottleneck. These limitations occur because raw sensor data is simplified into a small set of geometric categories, which can remove important hidden information needed to understand complex traffic situations [4]. The approach presented in this thesis uses the same spatial data but avoids these bottlenecks by transforming the information into natural language-like sequences. The planner still uses the same safety and cost-minimization logic, but it now operates on more detailed input that captures contextual aspects of the environment, such as observed driver behaviors and situational context.

Symbolic planners are typically limited by the input they receive, which describes only the current state of the environment [4]. They lack access to predictions about how other agents will behave, limiting their ability to anticipate future conflicts. The approach presented in this thesis augments the A* planner with LLM-generated predictions of future agent behavior, enabling it to reason about potential future interactions while preserving its core planning logic. This limitation becomes particularly critical in dynamic environments such as intersections, where safe decision-making depends on anticipating the future behavior of multiple interacting agents.

2.3 Neuro-symbolic approaches

Recent research has explored the integration of neural and symbolic methods to combine the strength of both paradigms [9]. Neuro-symbolic AI aims to leverage the flexibility and learning capabilities of neural networks together with the interpretability and logical reasoning of symbolic systems.

One example of such an approach is the framework presented in [6], which introduces a multi-stage pipeline combining generative and formal methods. In this architecture, an LLM serves as a proposal engine that translates environmental observations into formal task descriptions. The authors utilize a Sense-Plan-Code-Act (SPCA) loop, where the core strength lies in its two-step verification. First, a symbolic planner confirms that the LLM’s plan is logically sound. Then, a second LLM stage translates that plan into machine code. This setup uses formal rules to guide the AI’s creativity and reduce the risk of generating invalid actions.

Another recent approach, SanDRA, also implements a decision-making framework combining LLM-based reasoning with formal verification [8]. This approach incorporates an LLM for the generation of driving actions and adds a formal verification layer. Unlike traditional methods that primarily formalize traffic rules, SanDRA systematically translates both LLM’s high-level actions (e.g. “accelerate” or “stop”) and traffic regulations into Linear Temporal Logic (LTL) formulas [8]. These formulas are then combined into a joint specification, allowing the reachability analysis to verify if a physically feasible trajectory exists that satisfies both the intended driving behavior and all legal safety constraints.

Integration enables the system to filter out unsafe or infeasible actions by verifying them against vehicle dynamics and formally specified traffic rules. This method provides stronger safety guarantees compared to the LLM-based methods alone. By combining LLM-based reasoning with formal verification, the approach ensures

that generated actions are both contextually meaningful and compliant with safety constraints [8].

The contribution of this thesis differs from both SPCA and SanDRA in both its objective and integration strategy. In the SPCA framework, symbolic reasoning is primarily used to verify and execute high-level plans generated by an LLM. In SanDRA, formal verification is used to constrain LLM-generated actions. In contrast, the approach presented in this thesis uses the symbolic planner as the primary decision-making component. Rather than generating or constraining high-level decisions, the LLM is used to predict the future behavior of surrounding agents. These predictions are incorporated into the planning process as enhanced input, allowing the planner to reason about potential future interactions while preserving its existing safety and decision-making logic.

3

Theoretical Background

This chapter discusses the theoretical foundation for neuro-symbolic AI.

3.1 Large Language models

This section provides essential background on LLMs. LLMs are advanced artificial intelligence models that are extensively unsupervised pre-training on huge text datasets to understand, generate, and handle human-like text. They are able to write articles, summarize texts, translate languages, and create contents like images or code [10].

3.1.1 Terminology

Model Size: The term "model size" refers to the number of parameters the model has [11]. Larger models typically have more parameters, enabling them to encode more data patterns and navigate complex language relationships. While this allows a larger model to capture more subtleties in the data, it's important to note that these models require significant computational resources for training and substantial memory to store the parameters.

Base model: A pre-trained language model provides the foundational layer from which specialized models are developed for diverse downstream tasks, such as translation or text classification [11].

Fine-tuning: As a specialized form of transfer learning, fine-tuning involves the further training of a pre-trained model on a targeted dataset to improve its performance for a specific domain or task [11].

Hallucinations: Refers to the phenomenon, where LLMs generate outputs that are either nonsensical or unfaithful to the provided source material. In these moments, the model generates false details that are grammatically correct but have no basis in fact [12].

3.2 Supervised Fine-Tuning (SFT)

Base models are typically pre-trained on large-scale unlabeled data and are therefore not inherently optimized for specific tasks [13]. Supervised fine-tuning adapts a

pre-trained model to a target task using labeled data. Model parameters are updated through gradient-based optimization. The goal is to minimize a task-specific loss function.

3.2.1 Full Fine-tuning

In full fine-tuning, all parameters of the pre-trained model are updated during training [13]. This method allows the model to fully adapt to the target task and can lead to strong performance, when the computational resources are sufficient.

In contrast, modern large language models often contains billions of parameters. This makes full fine-tuning computationally expensive in terms of GPU memory, storage, and training time. Furthermore, maintaining separate fully fine-tuned models for different tasks is inefficient. Each task requires storing a complete copy of the model.

3.2.2 Parameter-Efficient Fine-Tuning (PEFT)

To address the limitations of full fine-tuning, parameter-efficient fine-tuning (PEFT) methods have been introduced. Instead of updating all model parameters, PEFT modifies only a small subset of parameters, while the majority of the pre-trained model remains frozen. This significantly reduces computational cost and memory requirements, while maintaining competitive performance compared to full fine-tuning [13].

One widely used PEFT method is Low-Rank Adaptation (LoRA). Instead of directly modifying the original model weights, LoRA introduces trainable low-rank matrices into selected layers of the model [14]. During training, only these additional matrices are updated, making the fine-tuning process much more efficient.

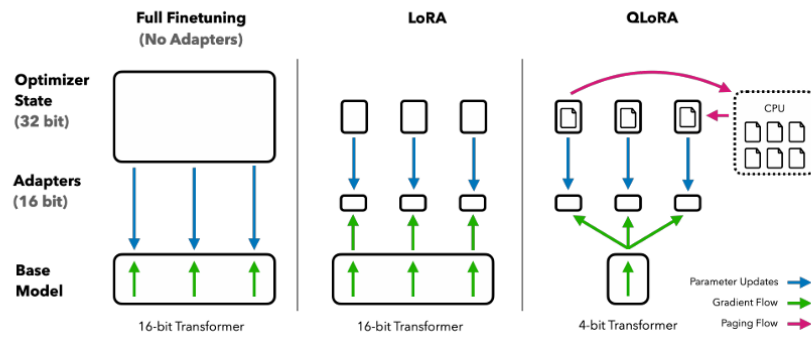


Figure 3.1: Comparison of different fine-tuning methods and their memory requirements. QLoRA reduces memory consumption compared to LoRA by quantizing the pre-trained model weights to 4-bit precision and using paged optimizers to handle memory spikes. Source: Adapted from [15].

For this thesis, Quantized Low-Rank Adaptation (QLoRA) was used due to limited computational resources. QLoRA extends LoRA by combining parameter-efficient

fine-tuning with model quantization [15]. Specifically, the pre-trained model weights are quantized to 4-bit precision, substantially reducing memory usage, while the low-rank adaptation matrices remain trainable in higher precision [15]. Figure 3.1 illustrates the differences between full fine-tuning, LoRA, and QLoRA in terms of parameter updates and memory requirements.

3.3 Symbolic Planner

This section presents the theoretical background necessary to understand how symbolic planners work, with a particular focus on the A* algorithm, which is used as the symbolic planner in this thesis.

3.3.1 Search-Based Path Planning

Autonomous driving requires the generation of safe and feasible trajectories under dynamic and uncertain environmental conditions. Among classical motion planning paradigms, search-based planning methods are highly relevant due to their ability to systematically explore feasible motion alternatives. This method optimizes predefined cost functions under operational constraints such as obstacle avoidance, vehicle dynamics, and traffic regulations [4].

In search-based planning, the motion planning problem is formulated as a state-space search problem. Nodes represent possible vehicle states and edges correspond to feasible transitions between those states. Different search strategies involve trade-offs between computational efficiency and solution quality. Cost-based exploration methods can provide optimal solutions but may become computationally expensive in large search spaces. In contrast, heuristic-driven approaches improve efficiency by directing exploration toward promising regions of the search space, though this may sacrifice optimality [16].

These trade-offs are particularly important in autonomous driving, where planning must satisfy both strict real-time requirements and safety constraints. This motivates the use of A* search, which balances computational efficiency with solution quality through heuristic-guided search.

3.3.2 A* Search Algorithm

A* is a heuristic search algorithm widely used for route planning in robotics and autonomous driving due to its balance between computational efficiency and solution quality [4]. As discussed in the previous section, search-based planning methods involve a trade-off between exhaustive exploration and heuristic efficiency. A* addresses this trade-off by combining exact path cost accumulation with heuristic goal-directed search [17].

The A* algorithm evaluates candidate nodes using the function:

$$f(n) = g(n) + h(n)$$

3. Theoretical Background

Where $f(n)$ represents the estimated total cost of a path passing through node n , $g(n)$ denotes the accumulated cost from the initial state to node n , and $h(n)$ represents a heuristic estimate of the remaining cost from node n to the goal state.

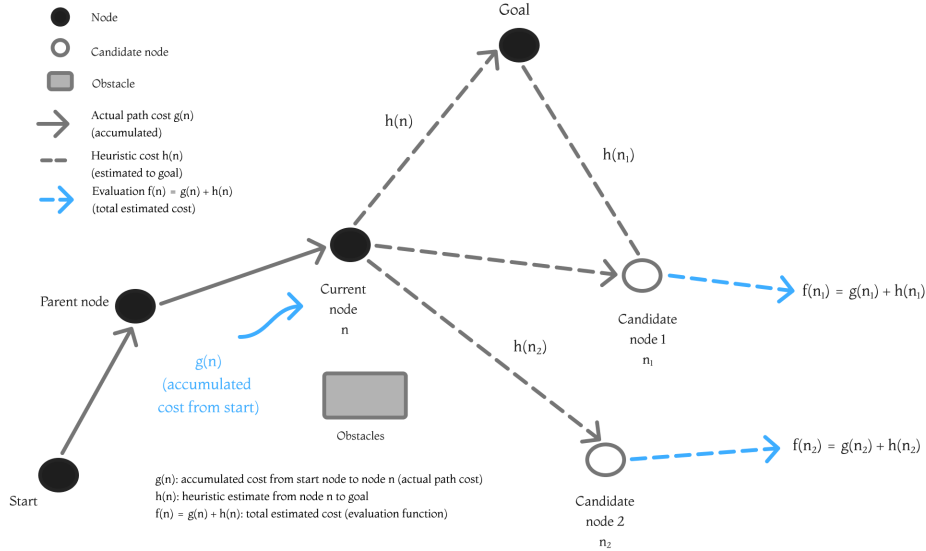


Figure 3.2: Illustration of the A* search principle. Candidate nodes are evaluated using the total estimated cost $f(n) = g(n) + h(n)$, where the node with the lowest evaluation value is selected for further exploration.

At each step, A* prioritizes the nodes with lowest estimated total cost for further exploration. In this way, the algorithm balances the cost of the path already traveled with an estimate of the remaining distance to the goal [17].

The relationship between $g(n)$ and $h(n)$ influences the behavior of the algorithm. When the heuristic provides little useful information about the remaining cost to the goal, A* behaves more similarly to Dijkstra's algorithm and explores a larger search space, increasing the computational cost. In contrast, an informative heuristic can more directly guide the search towards the goal and, thus, reduce the search space. However, the influence of the heuristic should not be confused with the guarantee of optimality of A*. Standard A* is guaranteed to find an optimal solution when the heuristic is admissible, meaning it never overestimates the true cost from a node to the goal. If A* heuristic is given additional weight then the search becomes more goal-oriented but may optimally perform worse for reduced computational cost. Greedy Best-First Search represents a more extreme case in which the search is guided only by $h(n)$ and does not account for the cost already accumulated. Therefore, the choice and properties of the heuristic function play an important role in balancing search efficiency and solution optimality for A*.

A common heuristic for spatial path planning is the Euclidean distance between the current node and the goal:

$$h(n) = \sqrt{(x_n - x_g)^2 + (y_n - y_g)^2}$$

where (x_n, y_n) represents the current node position and (x_g, y_g) the goal position. When the movement costs are based on Euclidean distances, this heuristic is admissible and consistent, since the straight-line distance never exceeds the length of a path between two points.

4

Methods

This chapter details the development and evaluation of the proposed neuro-symbolic framework. The methodology is divided into four main stages, scenario design and data gathering, LLM trajectory prediction, symbolic path planning, and evaluation.

4.1 System Overview

The framework combines an LLM with symbolic planning to integrate semantic reasoning with formal safety guarantees. Rather than using the LLM as an end-to-end decision-maker, the system separates trajectory prediction from path planning. The LLM predicts surrounding actors future trajectories, and the purpose of the symbolic A* planner is to use these predictions to generate a safe, collision-free trajectory for the ego vehicle.

The pipeline consists of two parallel approaches that are evaluated for comparison. In the neuro-symbolic approach, a fine-tuned LLM predicts the future positions of surrounding actors. These predictions are incorporated into a grid-based occupancy representation and used by the A* planner to reason about spatial-temporal safety. In the end-to-end baseline approach, the same pre-trained LLM is separately fine-tuned to directly generate an ego vehicle trajectory without symbolic verification, providing a control for evaluating the benefit of explicit safety planning.

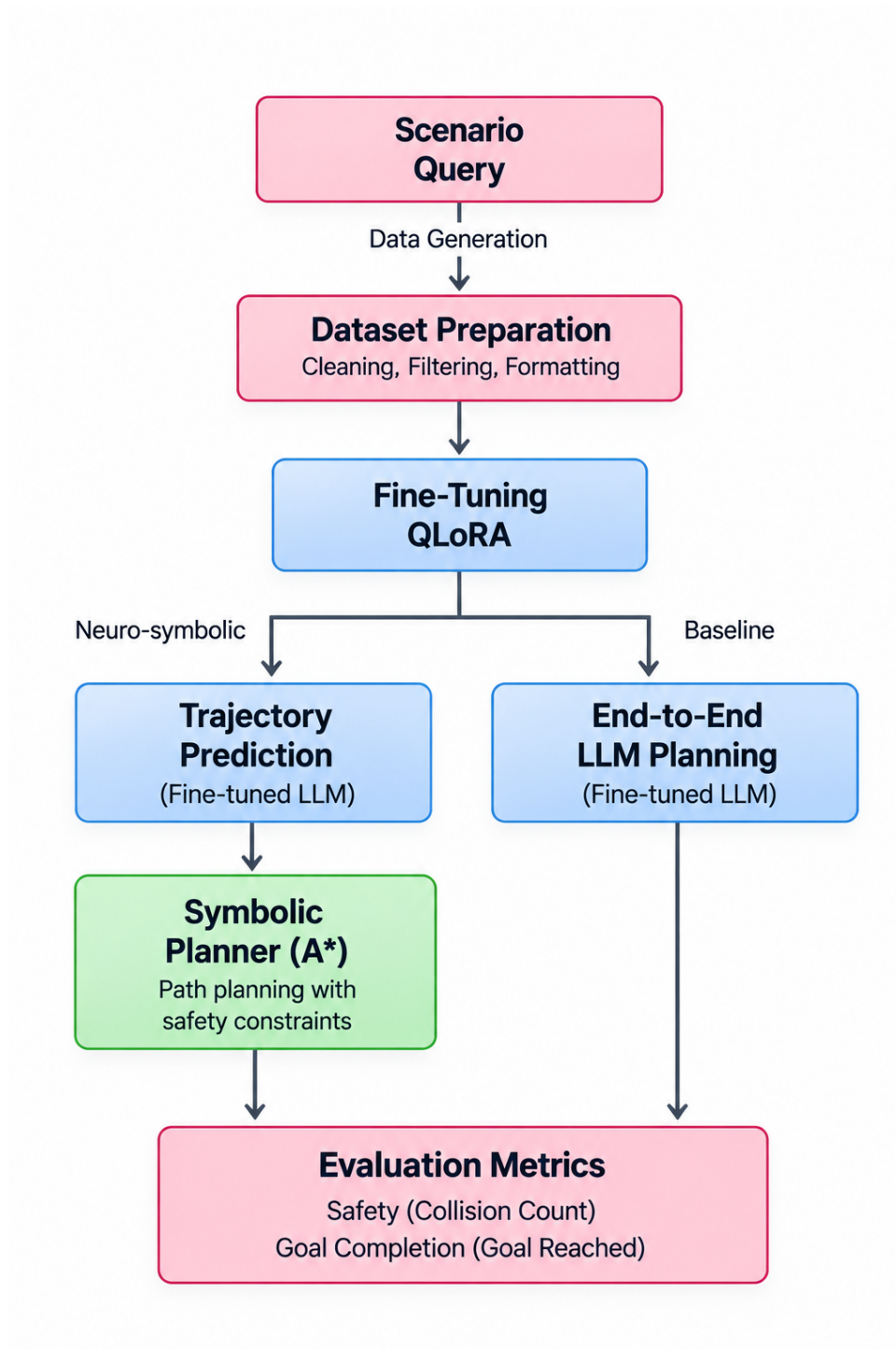


Figure 4.1: Overview of the proposed pipeline, including scenario generation, dataset preparation, QLoRA fine-tuning, and the evaluation of the neuro-symbolic and end-to-end planning approaches based on safety and goal completion.

Data flows through four main stages: (1) scenario generation, in Scenic, which are then executed in CARLA producing simulated driving scenes with varying actors, speeds and distances, (2) pre-processing converts raw JSON logs from the simulated driving scenes into natural language-formatted training sequences that

contain measurements of the last five steps, (3) the Mistral 3-8B model is fine-tuned with QLoRA on the same dataset for trajectory prediction and end-to-end planning, but only the safe scenarios are used for training the end-to-end planner. For the neuro-symbolic approach, the fine-tuned LLM predicts future trajectories, which are then integrated into a symbolic A* planner to generate collision-free paths towards the goal, (4) both the fine-tuned end-to-end planner and the neuro-symbolic planner, which combines LLM-based trajectory prediction with A* search, are evaluated using the safety metrics of collision rate and goal completion. The comparative evaluation directly addresses whether integrating LLM-based trajectory prediction with symbolic A* search improves safety relative to pure LLM-based planning. Figure 4.1 illustrates a overview of the proposed system architecture.

4.2 Dataset Generation

There are two parts to obtaining a dataset in this thesis project. The first is to create a scenario in the Scenic language that defines ranges of parameter values rather than fixed values, allowing each simulation run in CARLA to produce a different variation of the same base scenario. The second is to collect a decided amount of data of that defined scenario.

4.2.1 Scenario Creation

Before collecting the data, the scenarios to be used for training the LLM had to be defined. One of the most common traffic scenarios that exist is the intersection scenario. An intersection scenario can have a wide variety of outcomes, since multiple factors affect the traffic behavior. The scenario can vary in the number of traffic participants, their directions of approach, and their speeds. Hence, for this thesis project the specific scenario chosen was a four-way intersection, because of its variability.

```

1 EGO_MODEL = "vehicle.tesla.cybertruck"
2 EGO_SPEED = Range(5, 10)
3
4 CARS_MODEL = "vehicle.audi.tt"
5 CARS_SPEED = Range(5, 10)
6
7 PEDESTRIAN_SPEED = Range(0, 3)

```

Listing 4.1: Example of speed range definitions for actors in a Scenic scenario.

```

1 pedestrian_one = new Pedestrian right of crossing_one by Range(1, 4),
2   facing Range(60, 120) deg relative to crossing_one.heading,
3   with regionContainedIn None,
4   with behavior PEDESTRIAN_Behavior(PEDESTRIAN_SPEED)
5 require 0 <= (distance from pedestrian_one to crossing_one) <= 20

```

Listing 4.2: Example of distance and heading definitions for actors in a Scenic scenario.

The language, Scenic, was used to develop the intersection scenarios. The three Scenic files created contain varying numbers and types of actors. In this context, an actor refers to either a pedestrian, a cyclist, or a vehicle. The first file includes an ego vehicle, a pedestrian, a cyclist, and another vehicle. The second file has an ego vehicle, a cyclist, and another vehicle. Lastly, the third Scenic file contains an ego vehicle, two other vehicles, and two pedestrians. All these actors traverse the intersection in some way, and by applying numerical ranges, multiple variations of the same scenario can be achieved. Listing 4.1 shows how a range is utilized to define different speeds for the actors. When the scenario is executed in the CARLA simulator, the speed values are randomly chosen from the specified range. The same technique is used to define the actors’ distances and headings, which can be seen in Listing 4.2.

4.2.2 Data Collection

For data collection, the CARLA simulation environment was used to run the defined Scenic scenarios. Each Scenic scenario was configured to run **2000** times. However, the CARLA simulations did not complete all **2000** iterations for any of the three defined scenarios. In total, **4382** JSON files were collected, where one JSON file captures one full scenario.

The data generated from each CARLA simulation run is stored in a JSON format, capturing the state of the environment at each timestep. The simulation is executed with a fixed timestep of 0.1 seconds and each simulation was set to a duration of 20 seconds. By choosing a timestep of 0.1 seconds, it is possible to gather significantly more data than with a timestep of e.g. 1.0 seconds. This higher temporal resolution provides more data points per scenario, giving the preprocessing step more flexibility in selecting which timesteps to include in the training samples. Each scenario’s JSON file contains metadata, such as the random seed, map, initial ego speed, and the timesteps. For every timestep, at an interval of 0.1 seconds, the system records its actors with a unique ID, vehicle type, spatial coordinates (x, y) , speed, and heading. Additionally, it records interaction metrics between the ego vehicle and the other actors, specifically the euclidean distance and time-to-collision (TTC).

4.3 A Neuro-Symbolic Planner

The proposed thesis approach combines an LLM-based trajectory prediction component with an A* planner. The LLM predicts the future trajectories of the surrounding actors, while the A* planner utilizes these predictions to plan a path for the ego vehicle.

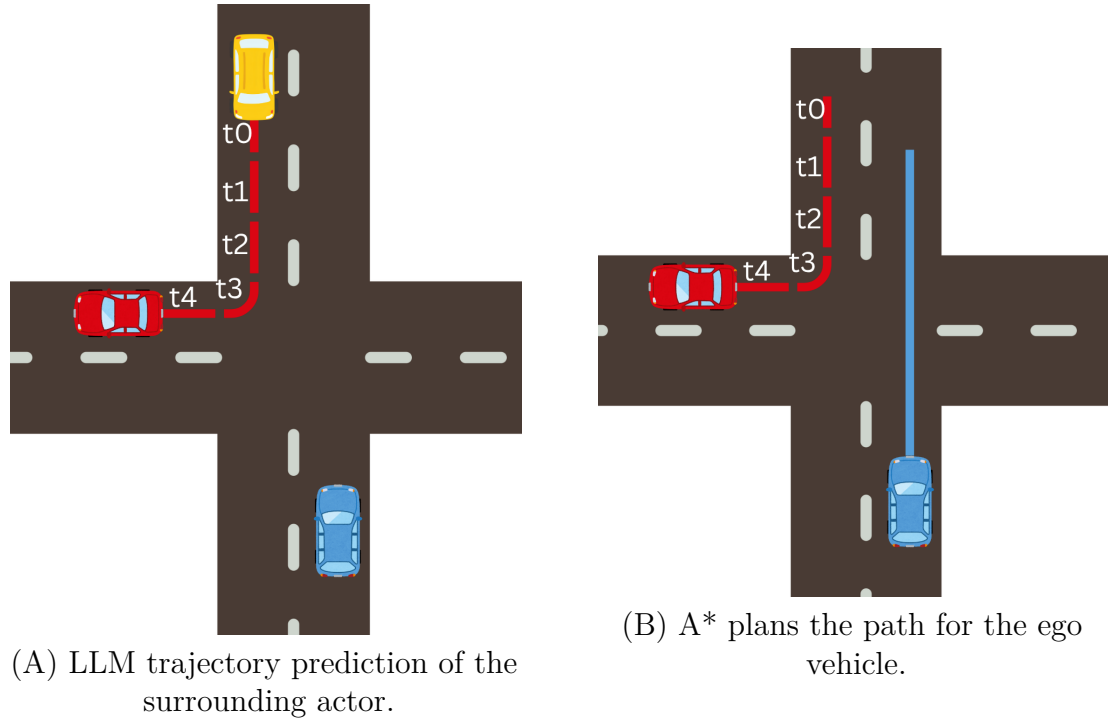


Figure 4.2: Overview of the proposed neuro-symbolic planner.

An overview of the proposed neuro-symbolic planner approach is shown in Figure 4.2. Part (A) of Figure 4.2 illustrates the neural component of the approach. The blue vehicle represents the ego vehicle, while the yellow vehicle represents the current position of a surrounding actor. The five red dashed lines represent the LLM’s predicted trajectory steps over the next five steps for that actor. Next, part (B) of Figure 4.2 illustrates the symbolic component of the approach. Here, the A* planner utilizes the LLM’s predicted trajectory of the surrounding actor over the next five time steps to determine a path, the blue line, for the ego vehicle.

4.3.1 An LLM for Predicting the Trajectories of Surrounding Actors

The *neural* component of the framework focuses on predicting the five future states of the surrounding actors. To achieve this, data preprocessing is applied before fine-tuning a pre-trained LLM.

4.3.1.1 Model Selection and Preparation

A pre-trained LLM, based on the Mistral architecture, is used as the base model. The model, Mistral-3-8B-Base-2512, is loaded with its corresponding tokenizer to ensure consistency between text representation and model input.

One issue encountered when retrieving the LLM was the company’s proxy, which blocked direct access to Hugging Face. As a result, the model had to be downloaded locally and then manually transferred to the Linux server for deployment.

Furthermore, *mistral-common*, a set of utilities for working with Mistral AI models, is required to enable compatibility with the Transformers library. This package also had to be downloaded locally and manually transferred to the server before use.

4.3.1.2 Data Preprocessing

The data generated from the CARLA simulator, described in Section 4.2, is stored in a JSON format. This data is converted into a JSONL format, where each line represents a single training sample for the LLM. The JSONL format was chosen over the standard JSON format because JSONL is designed to handle larger and more scalable datasets, which is an important factor for LLM usage [18].

The input data for the LLM is further transformed into a more textual format by incorporating linking words, such as *with* and *and*. The idea is to improve the LLM’s contextual understanding of the environment by providing coherent sentences rather than isolated lines with data. There are two lines with the abbreviation TTC, time to collision, where the number of TTC lines correspond to the number of actors present in the input. The input data is structured as follows:

```

1 "PROMPT: ":
2 Trajectory history (5 steps)
3 t: seconds
4 Ego: vehicle type (vehicle id) with position (x, y), has speed m/s and
   heading degrees
5 Other: vehicle type (vehicle id) with position (x, y), has speed m/s and
   heading degrees
6 Other: vehicle type (vehicle id) with position (x, y), has speed m/s and
   heading degrees
7 Distance: meters, TTC: seconds
8 Distance: meters, TTC: seconds

```

Thus, the data is reformulated into more natural language-like sequences. These descriptions capture the states of all actors in the scene, including the ego vehicle. By representing the data in this textual format, the model can process it as a sequence and learn to predict future states. The prompt component follows the same format across all samples, which makes it easier for the model to learn patterns in the data. Since the input contains exactly five time steps, the task can be viewed as a time-series prediction problem. In this setting, the model can learn how past states and interactions between actors influence future behavior.

The ground truth data, representing the five future time steps, is stored in a similar JSONL format. Its structure is very important, as it determines the format of the LLM’s output, which consists of predicted trajectories for the surrounding actors over the next five time steps. The LLM is tasked exclusively with predicting the future trajectories of actors surrounding the ego vehicle. Therefore, only these actors are included in the ground truth data. The format of the ground truth is as follows:

```

1 "GROUND TRUTH: ":
2 "id: <actor_id1>, pos: (x, y); id: <actor_id2>, pos: (x, y)\n
3 id: <actor_id1>, pos: (x, y); id: <actor_id2>, pos: (x, y)\n
4 id: <actor_id1>, pos: (x, y); id: <actor_id2>, pos: (x, y)\n
5 id: <actor_id1>, pos: (x, y); id: <actor_id2>, pos: (x, y)\n
6 id: <actor_id1>, pos: (x, y); id: <actor_id2>, pos: (x, y)"

```

4.3.1.3 Fine-Tuning the LLM

Before fine-tuning the LLM, the collected dataset was divided into training, validation, and test sets. Before dividing the collected dataset, duplicate JSON files were identified and removed to ensure that each scenario was represented only once. Additionally, since the 0.1 second time steps gathered from the CARLA simulator resulted in only small movements of the actors, the data was downsampled to every 0.5 seconds. The dataset was then divided into training, validation and test sets. The **4382** collected scenario logs, mentioned in Section 4.2.2, were randomly split at log-level, with 80% allocated to the training set, and 10% each to the validation and test sets. Each scenario log was assigned to a single set before the training examples were generated. This guaranteed that examples originating from the same scenario log did not appear across dataset splits, thus preventing potential data leakage between the sets.

After dataset preparation, we applied parameter-efficient fine-tuning to the pre-trained language model using QLoRA (Quantized Low-Rank Adaptation). This approach enables efficient training by reducing memory usage while maintaining strong performance.

First, the base model is prepared for low-bit (k-bit) training, allowing it to operate with reduced precision. The original model parameters are kept frozen during training. Instead of updating all model parameters, LoRA introduces additional trainable parameters in the form of low-rank matrices that are incorporated into the model’s linear layers.

The LoRA configuration is controlled by three main hyperparameters: (1) the rank r , which determines the dimensionality of the low-rank matrices, (2) the scaling factor α , which controls the contribution of the LoRA update to the original weights, (3) the dropout rate, which regularizes the LoRA layers during training. In this work, $r = 8$, $\alpha = 32$, and the dropout rate is 0.1. The adaptations are applied to all linear layers in the model. This significantly reduces the number of trainable parameters compared to full fine-tuning.

The model is trained using a causal language modeling objective, in which it learns to predict the next token in a sequence. The input text is tokenized and truncated to a maximum length of 1024 tokens. Padding is applied where necessary, and padding tokens are ignored during loss computation. Training is performed using the Hugging Face Trainer framework. A per-device batch size of 1 was used to accommodate the model within the available GPU memory. Gradients were accumulated over 4 steps, resulting in an effective batch size of 4. A learning rate of 2×10^{-4} was

used for training. A maximum of 3 epochs was set based on previous training runs, where additional training did not produce any meaningful improvement in model performance. To reduce the risk of overfitting, the validation loss was evaluated at every 200 training steps, with early stopping enforced after three consecutive evaluations without improvement. Checkpoints were saved based on validation loss, and the checkpoint with the lowest validation loss was saved at the end of training and used as the final model.

Parameter	Value
Fine-tuning method	QLoRA
LoRA rank (r)	8
LoRA scaling factor (α)	32
LoRA dropout	0.1
Target modules	All linear layers
Training objective	Causal language modeling
Maximum sequence length	1024 tokens
Batch size per device	1
Gradient accumulation steps	4
Effective batch size	4
Number of epochs	3
Learning rate	2×10^{-4}
Base model parameters	Frozen
Trainable parameters	LoRA adapters only
Padding in loss	Ignored
Training framework	Hugging Face Trainer

Table 4.1: QLoRA fine-tuning configuration for the LLM.

During training, only the LoRA parameters are updated, while the original base model weights remain unchanged. After training, the fine-tuned LoRA adapters and tokenizer are saved for later use in trajectory prediction. As a result, only the adapter parameters need to be stored, rather than a complete copy of the entire model. The full QLoRA fine-tuning configuration for the LLM can be viewed in Table 4.1.

4.3.1.4 Evaluation Metrics

To evaluate the trajectory prediction accuracy, we employ the Average Displacement Error (ADE) metric. ADE is defined as the mean Euclidean distance between the predicted positions and the ground truth positions across all prediction timesteps:

$$\text{ADE} = \frac{1}{T} \sum_{t=1}^T \sqrt{(\hat{x}_t - x_t)^2 + (\hat{y}_t - y_t)^2}$$

where $\hat{x}_t, \hat{y}_t$ are the predicted coordinates at timestep t , x_t, y_t are the ground truth coordinates, and T is the prediction horizon.

4.3.2 A* Planner

The *symbolic* component of the framework is an A* search planner. The planner uses the predicted trajectories of surrounding actors given by the fine-tuned LLM to evaluate potential collisions and identify a safe, collision-free, trajectory through the scenario for the ego vehicle.

4.3.2.1 Environment Representation

The driving environment is represented as a local ego-centric occupancy grid. This grid provides a discrete representation of the area surrounding the ego vehicle, which allows the planner to reason about free space and obstacles in a structured way. The grid is centered on the ego vehicle and covers only a limited local region of the entire CARLA map. This technique reduces computational complexity and prioritizes the areas most relevant to planning.

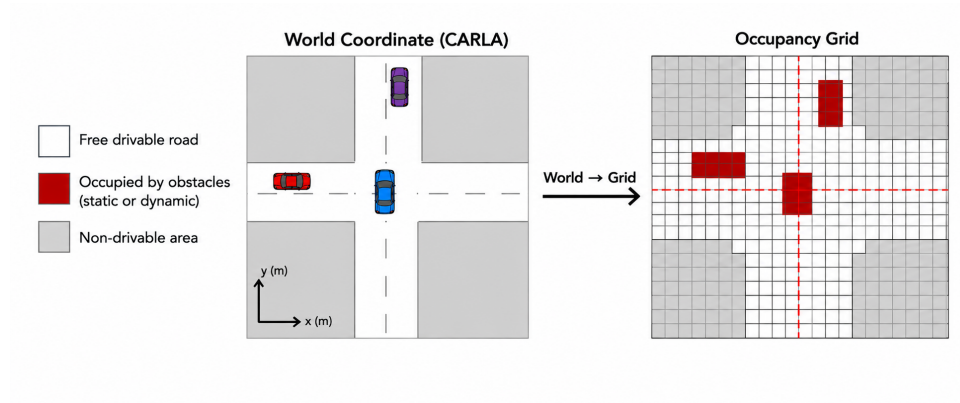


Figure 4.3: Example of local ego-centric occupancy grid.

The grid uses a spatial resolution of 0.5 meters per cell, meaning that each grid cell represents a physical area of 0.5×0.5 meters in the simulation environment. The planning range extends 50 meters in each direction from the ego vehicle, resulting in a total area of $100 \text{ m} \times 100 \text{ m}$, which at a resolution of 0.5 m per cell produces a grid of 200×200 cells. To transform CARLA world coordinates into grid coordinates, a coordinate transformation is applied relative to the ego vehicle position. The transformation is defined as follows:

$$g_x = \frac{x - x_{\text{ego}} + R}{\text{resolution}} \quad (4.1)$$

$$g_y = \frac{y - y_{\text{ego}} + R}{\text{resolution}} \quad (4.2)$$

Symbol	Description
x, y	World coordinates
$x_{\text{ego}}, y_{\text{ego}}$	Ego vehicle coordinates
R	Local planning range (50 m)
resolution	Grid cell size (0.5 m)

Table 4.2: Parameters used for coordinate transformation

The parameters used in the coordinate transformation are summarized in Table 4.2. This transformation maps continuous CARLA world coordinates to discrete grid indices, as seen in Figure 4.3.

Value	Description
0	Free drivable road
1	Occupied by predicted dynamic obstacles
2	Non-drivable area (off-road regions)

Table 4.3: Occupancy grid cell states

The occupancy grid uses three cell states to represent the environment, as shown in Table 4.3. A value of 0 represents a free drivable road, a value of 1 represents an occupied cell, and a value of 2 represent a non-drivable area. These states allow the A* planner to distinguish between traversable and unsafe regions in the map while accounting for the predicted positions of the surrounding actors.

4.3.2.2 Road Representation

The road geometry is obtained directly from the CARLA map by sampling road waypoints along the drivable road network. These sampled waypoints provide a discrete approximation of the geometry of the road center line in world coordinates. Instead of using the full global map during planning, only the road points located within the local planning region around the ego vehicle are considered.

To determine whether a grid cell belongs to the drivable road surface, the Euclidean distance from the center of the cell to the nearest sampled road waypoint is computed. Since the waypoints are densely sampled, this distance provides a close approximation of the lateral distance to the road center-line. A predefined lane half-width is used as the threshold, representing the maximum distance from the center-line at which a cell is considered part of the drivable corridor. If this distance is smaller than the predefined lane half-width, the cell is classified as drivable, otherwise, it is classified as non-drivable. In this way, the continuous road geometry from CARLA is converted into a local binary road mask.

Formally, the road mask $M(g_x, g_y)$ is defined as:

$$M(g_x, g_y) = \begin{cases} 1, & \text{if } \min_{(x_r, y_r) \in \mathcal{R}} \sqrt{(x_{cell} - x_r)^2 + (y_{cell} - y_r)^2} \leq w \\ 0, & \text{otherwise} \end{cases}$$

where $M(g_x, g_y) = 1$ denotes a drivable road cell and $M(g_x, g_y) = 0$ denotes a non-drivable cell. Furthermore, $\mathcal{R}$ denotes the set of sampled road waypoints, (x_r, y_r) is a road waypoint in world coordinates, (x_{cell}, y_{cell}) is the center of the current grid cell, and w is the road-width threshold. This road mask is then used to distinguish drivable road cells from off-road cells in the occupancy grid.

4.3.2.3 Trajectory Prediction Representation

The trajectory predictions generated by the LLM are incorporated into the planning framework as dynamic obstacles. These predictions are provided as future actor positions in CARLA world coordinates and must therefore be transformed into a representation that can be used by the occupancy-grid-based planner.

The world-coordinate positions are converted into local grid coordinates using the ego vehicle’s current position as the reference. The predicted trajectories preserve their temporal ordering and are used separately by the A* planner for time-indexed collision checking and the evaluation of future conflicts with surrounding actors.

The predicted trajectories consist of discrete position estimates associated with future timestamps. The A* planner uses these predictions to assess whether potential collisions at future planning steps could occur.

The predicted actor positions are projected onto the local occupancy grid using the coordinate transformation described in Equations 4.1 and 4.2. As a result, every actor is represented as a sequence of occupied grid cells over time. The predicted trajectory of actor i can therefore be represented as a sequence of grid-cell coordinates associated with timestamps:

$$O_i = \{(g_x^1, g_y^1, t_1), (g_x^2, g_y^2, t_2), \dots, (g_x^n, g_y^n, t_n)\}. \quad (4.3)$$

Here, (g_x^k, g_y^k) denotes the grid cell coordinates occupied by the actor at time t_k .

Unlike static obstacles, dynamic obstacles can change their position over time as the actor moves according to the predicted trajectory. This means that the planner must consider both where obstacles are located and when they will be there. By doing so, it can detect potential collisions in advance instead of only reacting when they occur. The resulting representation of dynamic obstacles is then used for collision checking and safety verification.

4.3.2.4 Motion Planning with A*

Once the driving environment and predicted actor trajectories have been represented in the occupancy grid, motion planning is performed using the A* search algorithm. The goal of the planner is to find a low-cost and safe trajectory from the current

ego vehicle’s position to a predefined goal position while avoiding both static and dynamic obstacles.

To account for the movement of surrounding actors, the planner incorporates both spatial and temporal information into the search process. The search is performed in a three-dimensional state space consisting of spatial position and time. A state is defined as

$$s = (x, y, t) \quad (4.4)$$

where x and y denote the occupancy grid coordinates and t denotes the planning timestep. Including time as part of the state representation enables the planner to reason about dynamic obstacles whose positions change over time. This means two states with identical spatial coordinates but different timestamps are considered different states. This allows the planner to evaluate, whether a particular location is safe to occupancy at a specific time.

The initial state corresponds to the current grid position of the ego vehicle, while the goal state is defined as the grid cell corresponding to the target location. The search process explores feasible state transitions until an appropriate path to the goal is found.

At each planning step, the ego vehicle may execute one of five discrete actions:

$$A = \text{up, down, left, right, wait} \quad (4.5)$$

The four movement actions move the ego vehicle one grid cell (0.5 m) in the corresponding direction on the occupancy grid, *up*, *down*, *left*, and *right* each shift the ego by one cell along the respective grid axis, and the wait action allows it to remain at its current position. These are discrete grid-level movements, not physical steering commands. The planned sequence of grid cells is then passed to CARLA’s built-in waypoint follower, which handles smooth vehicle control.

The wait action is important in dynamic environments because it allows the vehicle to stay in the same position when a future collision is predicted. Instead of taking an unnecessary detour, the vehicle can simply wait until the other actor has passed and the path is safe again.

Each action has a unit cost and advances the planning process by one timestep. The transition model defines how the planner moves from one state to another after executing an action. Given a current state

$$s = (x, y, t) \quad (4.6)$$

and an action defined by the displacement

$$a = (\Delta x, \Delta y), \quad (4.7)$$

the resulting successor state is computed as

$$T(s, a) = (x + \Delta x, y + \Delta y, t + 1). \quad (4.8)$$

Action	Δx	Δy
up	0	+1
down	0	-1
left	-1	0
right	+1	0
wait	0	0

Table 4.4: Grid displacement for each action.

The displacement values $(\Delta x, \Delta y)$ for each action are fixed as shown in Table 4.4. Furthermore, every action advances the planning time by one timestep regardless of whether the ego vehicle moves or waits.

Before a successor state is added to the search tree, a number of checks are performed. States outside the occupancy grid are removed, and states that move into non-drivable cells are rejected. The planner also checks for collisions with the predicted trajectories of surrounding actors. Only states that pass all checks are considered during planning.

By combining spatial reasoning with temporal reasoning, the transition model enables the planner to anticipate future conflicts and generate collision-free trajectories in dynamic traffic scenarios.

4.3.2.5 Safety Verification

To find a path towards the goal, the planner evaluates candidate states with respect to the predicted motion of surrounding actors. Safety verification is integrated directly into the A* planning process and is performed when successor states are evaluated. It consists of three complementary components, collision checking, a temporal safety buffer, and future conflict evaluation. Collision checking rejects states that violate the hard collision constraint, while the temporal safety buffer extends collision checking across a forward time window. Future conflict evaluation further discourages the planner from selecting states that may lead to unsafe interactions with surrounding actors in the near future.

Collision checking is performed whenever a new successor state is generated. The planner compares the position of the ego vehicle with the predicted positions of surrounding actors and computes the Manhattan distance:

$$d(s, o_i) = |x - p_x| + |y - p_y|$$

where (x, y) denotes the position of the ego vehicle and (p_x, p_y) denotes the predicted position of the actor i .

To account for the separation required between actors, a collision radius r_c is defined around each predicted actor position. A state is considered unsafe if $d(s, o_i) \leq r_c$. When the condition is satisfied, the state is rejected and not added to the search tree. hard constraint prevents the planner from considering states that bring the ego vehicle within the specified collision radius of a predicted actor position.

Although the collision check removes states where the ego vehicle would be too close to another vehicle at a given timestep, there is still a possible source of error. The planner’s timesteps and the LLM’s predicted timestamps do not always match exactly. In addition, the ego vehicle may not move exactly as planned because of delays or limits in acceleration. This means that the vehicle’s actual position can be different from the planned position, which may affect the collision check.

To compensate for this temporal uncertainty, a temporal safety buffer of size T_b is introduced. Instead of checking for a collision only at the exact current planning timestep t , the safety check is extended over a forward-looking time window:

$$\check{t} \in [t, \min(t + T_b, T_{\max})] \quad (4.9)$$

where $T_{\max}$ is the length of the prediction horizon. A state $s = (x, y, t)$ is therefore considered unsafe if any predicted actor i is located within the collision radius r_c at *any* timestep $\check{t}$ within this window:

$$\exists i, \exists \check{t} \in [t, \min(t + T_b, T_{\max})] : d((x, y), p_i(\check{t})) \leq r_c \quad (4.10)$$

where $p_i(\check{t}) = (p_x, p_y)$ is the predicted grid position of actor i at time $\check{t}$.

The temporal buffer is set to $T_b = 12$ planning steps, with each planning step corresponding to Δt_{plan} seconds. This corresponds to a forward-looking temporal buffer of $12\Delta t_{\text{plan}}$ seconds checking the current timestep and up to 12 subsequent planning steps. The actual window may be shorter near the end of the prediction horizon. This makes the collision check more conservative and reduces the risk of accepting states that seem safe at one exact time but may become unsafe because of timing differences.

Collision checking removes states that are immediately unsafe. A state can pass the collision check and still lead to an unsafe situation a short time later. To handle this, the planner also evaluates possible future conflicts. Rather than completely rejecting these states, a soft penalty is assigned to states to bring the ego vehicle too close to other actors within a short lookahead horizon. For a state $s = (x, y, t)$, the planner checks the predicted positions of all actors over a lookahead horizon of $H = 12$ planning steps, that is $t' \in \{t, t + 1, \dots, t + 11\}$. For each of these time steps and for each actor i , it calculates the Manhattan distance between the ego vehicle and the predicted position of the actor:

$$d_i(t') = |x - p_x^i(t')| + |y - p_y^i(t')| \quad (4.11)$$

If this distance falls below a safe distance threshold d_{safe} , a penalty proportional to the magnitude of the violation is incurred:

$$\pi_{\text{future}}(s) = w_f \sum_i \sum_{t'=t}^{t+H-1} \max(0, d_{\text{safe}} - d_i(t')) \quad (4.12)$$

where $w_f = 5$ is a fixed weighting factor, $d_{\text{safe}} = 6$ grid cells, and $H = 12$ planning steps. These parameter values were selected empirically through simulation testing. This mechanism discourages the planner from entering states that are likely to converge towards occupied cells in the near term, even when those states do not immediately trigger a hard collision constraint. In combination with the collision check and the temporal safety buffer, the future conflict evaluation produces a safety-aware planning behaviour that considers both present and near-future interactions.

4.3.2.6 Cost Function and Heuristic

The A* algorithm evaluates each state by combining the accumulated cost from the start state with a heuristic estimate of the remaining cost to the goal. This is expressed using the standard formulation:

$$f(s) = g(s) + h(s) \quad (4.13)$$

where $g(s)$ is the accumulated cost from the start state to s , and $h(s)$ is the heuristic estimate of the remaining cost to the goal. The function $f(s)$ therefore represents the estimated total cost of a path through state s .

The accumulated cost $g(s)$ is calculated by summing the transition costs along the path. The transition cost for applying action a in state s is defined as:

$$C(s, a) = c_{\text{move}}(a) + \pi_{\text{safe}}(s) + \pi_{\text{future}}(s) + \pi_{\text{route}}(s) \quad (4.14)$$

where $c_{\text{move}}(a)$ is the base movement cost, and $\pi_{\text{safe}}(s)$, $\pi_{\text{future}}(s)$, and $\pi_{\text{route}}(s)$ are penalty terms associated with safety, future conflicts, and route consistency, respectively.

The heuristic function $h(s)$ estimates the remaining cost from the current state to the goal and guides the search toward promising states. In this work, the heuristic is based on the Manhattan distance between the current position and the goal:

$$h(s) = |x_s - x_{\text{goal}}| + |y_s - y_{\text{goal}}| \quad (4.15)$$

For A* to guarantee an optimal solution, the heuristic must be admissible, meaning that it never overestimates the minimum remaining cost to the goal. This depends on the movement costs used by the planner and the relationship between those costs and the Manhattan distance.

The safety cost penalties states in which the ego vehicle is in close proximity to a predicted actor at the current timestep. As described in the collision-checking procedure in Section 4.3.2.5, states within the hard collision radius r_c are rejected

during planning. The soft safety penalty is additionally applied to states that are near but outside this hard boundary. This gradient penalty discourages the planner from planning trajectories that pass too close to surrounding actors, even when no hard collision is detected.

$$\pi_{\text{safe}}(s) = \max(0, d_{\text{safe}} - d_{\text{min}}(s)) \cdot w_s \quad (4.16)$$

where $d_{\text{min}}(s)$ is the minimum Manhattan distance to any predicted actor position within the temporal safety window, $d_{\text{safe}} = 6$ grid cells is the soft safety threshold, and $w_s = 2$ is the associated penalty weight. States in which $d_{\text{min}}(s) \geq d_{\text{safe}}$ receive no safety penalty, while states that are closer to the nearest actor receive a proportionally larger penalty.

The heuristic function provides a lower-bound estimate of the cost remaining from a given state to the goal.

$$h(s) = |x - g_x| + |y - g_y| \quad (4.17)$$

where (g_x, g_y) is the goal grid cell and (x, y) is the current ego grid position.

The Manhattan distance is used as an admissible heuristic since each action moves the ego vehicle by exactly one grid cell. Therefore, it never overestimates the true cost of reaching the goal, allowing A* to find an optimal path according to the defined cost function. In addition, the heuristic is consistent because it satisfies the triangle inequality for every state transition. As a result, A* avoids re-expanding closed nodes, improving the efficiency of the search.

4.4 An End-to-End Neural Planner

The end-to-end neural planner serves as a baseline for evaluating the proposed neuro-symbolic approach. Unlike the neuro-symbolic planner, the end-to-end planner directly generates the ego vehicle’s trajectory from the input without any other component.

4.4.1 Motivation and Design

The previous approach combines an LLM for trajectory prediction with a symbolic A* planner, where the predicted trajectories are represented as dynamic obstacles during planning. This modular design separates prediction and planning into two independent components.

To provide a baseline for comparison, we also develop an end-to-end planning approach based entirely on a fine-tuned LLM. Rather than predicting the future trajectories of surrounding actors, this model directly predicts the ego vehicle’s trajectory through the scenario. Consequently, the planning process is performed entirely by the LLM without relying on a separate symbolic planner.

The purpose of this model is not to replace the neuro-symbolic planner, but to enable a direct comparison between modular neuro-symbolic planning and end-to-end LLM-based planning under the same driving scenarios.

4.4.2 Dataset Generation

The end-to-end planner dataset was generated from the same original scenario JSON logs used for the trajectory-prediction dataset. The original scenarios were generated with a randomness to actor behavior and did not define a specific goal for the scenario, since the scenarios were generated for the trajectory-prediction component. For the end-to-end planner, a goal and the ego vehicle's delta to the goal were added as additional input. The end-to-end planner dataset was restricted to safe scenarios where there occurred no collisions, resulting in fewer scenarios for training the end-to-end deep-learning planner. The supervision also differs between the two datasets. Rather than extracting the future positions of surrounding actors, the ego vehicle's own future behavior is extracted as the training target. Although the end-to-end planner was conditioned on the goal position and the ego vehicle's delta to the goal, the ego trajectories were not originally generated with an explicitly defined goal. Therefore, the dataset had to be adjusted so that the last position of the ego vehicle in the scenario became the goal position if the position required the ego vehicle to traverse through the intersection and stop at a safe position after it.

Each scenario is associated with a fixed position of the target (g_x, g_y) defined by the scenario family and stored in a goal configuration file. For each training sample, the current distance and direction to the goal are computed and included in the prompt, making the planning task goal-conditioned. As with the trajectory-prediction dataset, the raw simulation data is sampled from 0.1 s to 0.5 s intervals and a sliding-window approach is applied with a history of 5 steps and a prediction horizon of 5 future ego steps. The resulting dataset is split into training, validation, and test sets.

4.4.3 Dataset Format

The dataset used to fine-tune the end-to-end planner follows a similar structure to that of the neuro-symbolic planner. Each sample consists of a "Prompt", the input to the LLM, and a "Ground Truth", the output of the LLM. However, the content and format of these components differ slightly in the two approaches. The following sections describe the dataset format used for the end-to-end baseline planner.

4.4.3.1 Input format

The input prompt extends the trajectory-history format from the trajectory-prediction dataset with additional goal information. Specifically, the prompt includes the goal position in world coordinates, the label of the goal, the current vector from the ego to the goal (*goal delta*), and the Euclidean distance to the goal. The general prompt structure is:

```

1 "PROMPT : "
2 Task: Predict the ego vehicle plan for the next 5 sampled steps in CARLA
   Town05.
3 Scenario: <scenario name>
4 Map: Carla/Maps/Town05
5 Goal: (<gx>, <gy>) in world coordinates
6 Goal label: <label>
7 Current goal delta: (<dx>, <dy>)
8 Current goal distance: <d> m
9 Output format: one line per future ego step as (x, y).
10
11 Trajectory history (5 steps)
12 t: <time> s
13 Ego: vehicle <type> (id <id>) with position (x, y),
   has speed <v> m/s and heading <degrees>
14 Other: actor <type> (id <id>) with position (x, y),
   has speed <v> m/s and heading <degrees>
15 Distance: <d> m, TTC: <ttc> s
16
17

```

By including the goal delta and the target distance in every sample, the model receives explicit information about the remaining task and can learn to adjust its planning behavior accordingly as the ego vehicle approaches the goal.

4.4.3.2 Output Format

The ground truth, the output of the planner, consists of 5 future ego steps in the format of (x, y) positions. This structured format allows the model to be executed directly in the simulator.

4.4.4 Fine-Tuning of the End-to-End Planner

The end-to-end LLM planner is fine-tuned using the same QLoRA configuration as the trajectory-prediction model. The Ministral-3B base model is loaded using 4-bit quantization, while the original model parameters remain frozen. LoRA adapters with a rank of $r = 8$, a scaling factor of $\alpha = 32$, and a dropout rate of 0.1 are applied to all linear layers.

The model is trained using a causal language modeling objective. Prompt tokens are masked with the label value -100 , so that the loss is computed only over the ground truth planning tokens. The same training hyperparameters are used as for the trajectory-prediction model, including a learning rate of 2×10^{-4} , a per-device batch size of 1, gradient accumulation over 4 steps, an effective batch size of 4, a maximum sequence length of 1024 tokens, and 3 training epochs.

This consistent configuration allows the two approaches to be compared under the same fine-tuning conditions, while the end-to-end planner differs in its training objective and dataset.

The QLoRA configuration and training hyperparameters are identical to those used for the trajectory-prediction model (Table 4.1).

4.5 Testing of the Neuro-Symbolic and End-to-End Neural Planners

The proposed neuro-symbolic approach was evaluated on the test set scenarios. First, the surrounding actor trajectory predictions were generated for all test scenarios and stored locally. These precomputed predictions were then used as inputs for the subsequent planning evaluation. For each scenario, the corresponding trajectory predictions, ego vehicle position, and goal position were provided to the A* planner. The planner was executed using a rolling window approach to generate a five-step trajectory for the ego vehicle towards the defined goal position. The generated trajectories were then chained together to form the complete trajectory for each scenario and evaluated with three categories, which are safe with goal completion, safe without goal completion, and collision.

The end-to-end planner was tested using the same set of test scenarios as the proposed neuro-symbolic approach. For each scenario, the end-to-end LLM planner was prompted to generate a five-step ego vehicle trajectory. These trajectories were subsequently chained together to form the complete trajectory for each scenario, following the same procedure as the A* planner. The resulting ego vehicle trajectories from both the end-to-end planner and the neuro-symbolic planner could therefore be evaluated under the same conditions and evaluated according to the same three categories.

5

Results

5.1 LLM's Accuracy

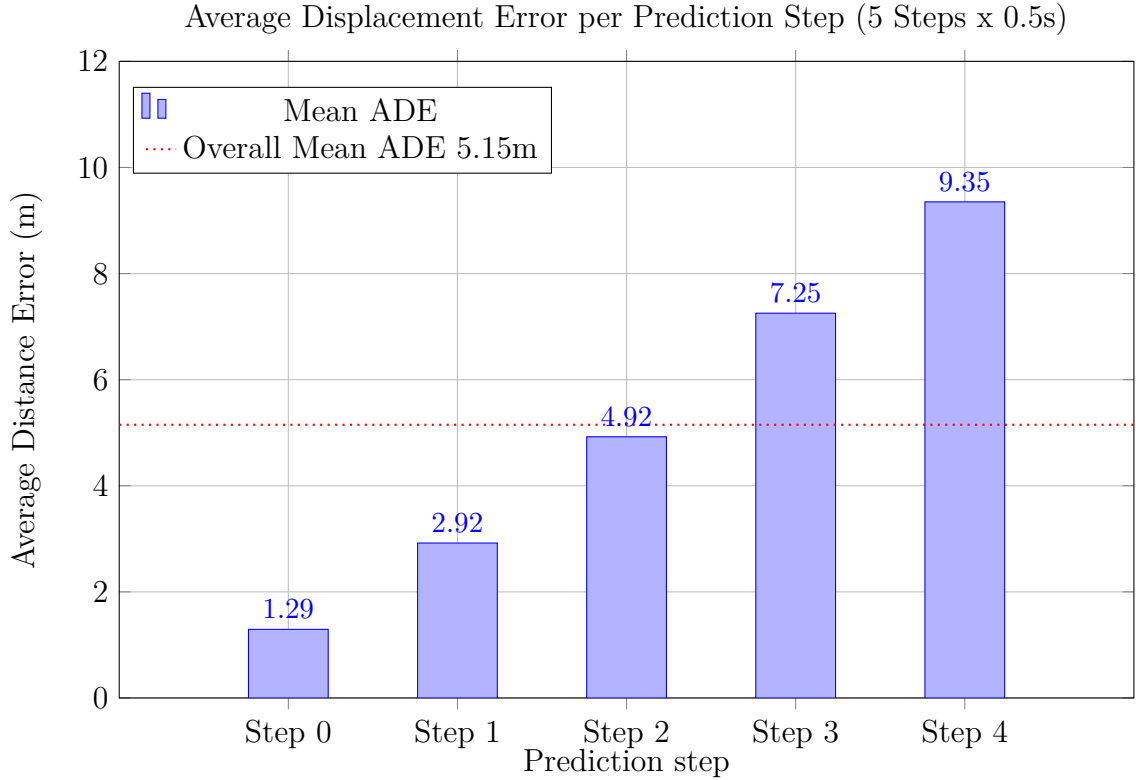


Figure 5.1: ADE across five prediction steps over a 2.5-second prediction horizon.

As shown in the Figure 5.1, the ADE increases as the prediction horizon becomes longer. The average distance error across all 1500 tests increases from 1.29 m at step 0 to 9.35 m at step 4, with an overall mean ADE of 5.15 m. These results show that the model is more accurate when predicting the vehicle's short term motion.

The lower error at the first prediction steps suggests that the model can capture the vehicle's immediate movement reasonably well, but the error becomes much larger at later steps. This means that the model is less accurate when predicting further into the future. This behaviour is expected because it is more difficult to predict the

vehicle’s future motion over longer time periods. Small errors in the early predictions can also build up over time and lead to larger position errors at later steps.

Overall, the model provides more reliable short-term trajectory predictions, while longer-term predictions are more difficult. This is important for downstream planning. Short-term predictions can support immediate vehicle control, while predictions further into the future should be treated with more uncertainty.

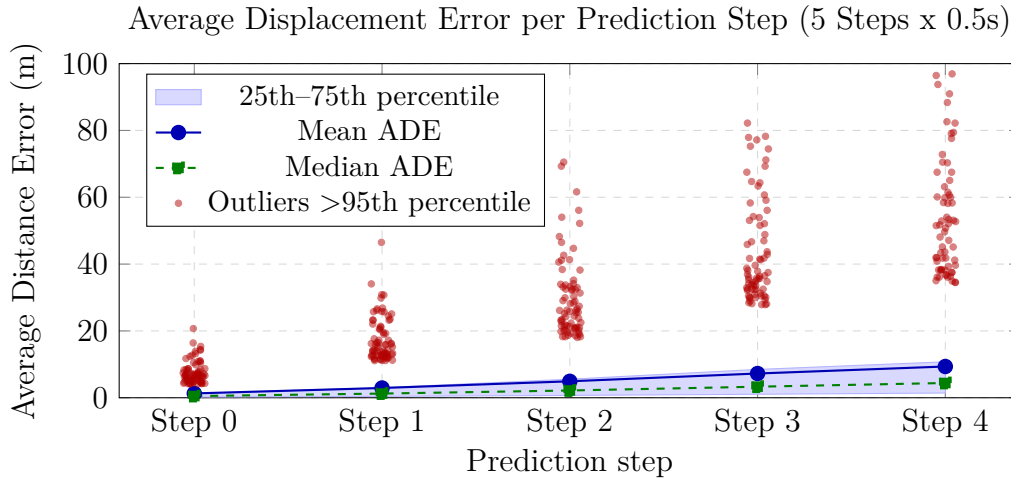


Figure 5.2: ADE across the five prediction steps, showing the mean, median, interquartile range, and high-error outliers.

Figure 5.2 provides a more detailed view of the displacement-error distribution across the prediction horizon by showing the mean and median (ADE), the interquartile range, and high-error outliers. Both the mean and median ADE increase progressively from Step 0 to Step 4, confirming that prediction accuracy decreases as the forecasting horizon becomes longer.

The interquartile range remains relatively close to the mean and median for the majority of samples, indicating that a substantial portion of the predictions has comparatively low displacement error. However, some predictions have much higher errors than others. These high-error cases become more common at later prediction steps, with some outliers exceeding 80–100 m at Step 4.

This shows that mean ADE alone does not fully describe the model’s performance. While the model performs reasonably well for many short- and medium-horizon predictions, some scenarios result in much larger errors. These cases may be caused by more uncertain situations or complex interactions with other actors and could be investigated further.

Overall, the results show that the model performs better for earlier prediction steps, while the prediction error increases for later steps. Although most predictions have relatively low errors, a small number of scenarios result in very large errors.

5.2 End-to-End Planner vs Neuro-Symbolic Planner

The main measurements of planner correctness are divided into three categories and are evaluated across 144 scenarios. The first category, **Safe & Reached Goal**, define the scenarios where the planner output an ego vehicle trajectory that was safe, meaning no collisions occurred, and reached the goal position. The second category, **Safe & Goal Not Reached**, define the scenarios where the planner output an ego vehicle trajectory which was safe, but did not reach the goal position. The last category, **Collisions** define the amount of scenarios where a collision occurred. A collision is when a position in the planner’s ego vehicle trajectory and one of the other actors ground truth positions in a scenario overlap.

Planner Type	Safe & Reached Goal	Safe & Goal Not Reached	Collisions
Neuro-symbolic	85 (59.0%)	12 (8.3%)	47 (32.6%)
End-to-end	39 (27.1%)	90 (62.5%)	15 (10.4%)

Table 5.1: Result of the neuro-symbolic planner in comparison to the end-to-end (baseline) planner over 144 simulations.

The most fundamental task of a planner, is to produce a path that is both safe and reach the goal destination. Table 5.1 shows that, for the category **Safe & Reached Goal**, the neuro-symbolic planner achieves a substantially higher success rate of 59.0% compared to only 27.1% for the end-to-end planner. This corresponds to an improvement by a factor of 2.1 in safe goal-reaching performance.

In Table 5.1 it is also shown how the end-to-end planner demonstrates a higher tendency to remain safe but is unable to reach the goal state, which accounts for 62.5% of the scenarios compared to 8.3% of the scenarios for the neuro-symbolic planner. Furthermore, it can be seen under **Collisions** in Table 5.1 that the end-to-end planner has a collision rate of only 10.4% while the neuro-symbolic planner has a collision rate of 32.6%. Based on these results, the end-to-end planner seems to be safer than the neuro-symbolic planner, which contradicts the research question of whether the neuro-symbolic approach improves safety in comparison to the standard end-to-end deep-learning-based planner. Nonetheless, the end-to-end planner performs poorly in reaching the goal in comparison to the neuro-symbolic planner. These results will be presented more in depth down below.

5.3 End-to-End Planner

The end-to-end planner results are presented in greater detail below. Building on the main result from Table 5.1, we present the same categories, this time broken down by the types of scenarios used for testing. Additionally, the results for the

Safe & Goal Not Reached category in Table 5.1 are interpreted to explain why a large portion of the scenarios are safe but do not reach the goal. For a consistent comparison, the end-to-end planner and the neuro-symbolic planner are evaluated on the same 144 scenarios, where the movements of the other actors are the same between both approaches.

Scenario Type	Evaluated	Safe & Reached Goal	Safe & Goal Not Reached	Collisions
Scenario with actors car, cyclist, and pedestrian	63	15 (23.8%)	48 (76.2%)	0 (0.0%)
Scenario with actors car and cyclist	28	10 (35.7%)	18 (64.3%)	0 (0.0%)
Scenario with actors 2 cars and 2 pedestrians	53	14 (26.4%)	24 (45.3%)	15 (28.3%)
ALL	144	39 (27.1%)	90 (62.5%)	15 (10.4%)

Table 5.2: End-to-end planner result across different scenario types, showing the number and its percentage of evaluated scenarios that safely reached the goal, remained safe but did not reach the goal, or resulted in a collision.

Out of all 144 evaluated scenarios, 129 (89.6%) of the scenarios resulted in a safe trajectory for the ego vehicle, while 15 (10.4%) resulted in a collision. As shown in Table 5.2, the end-to-end planner’s results vary across different scenario types. The highest rate of safe goal completion was observed in scenarios with a car and cyclist, where 10 out of 28 scenarios (35.7%) safely reached the goal position. In scenarios involving a car, a cyclist, and a pedestrian, only 15 out of 63 scenarios (23.8%) safely reached the goal position. Lastly, the scenario type with two cars and two pedestrians achieved a safe goal-completion rate of (26.4%), with 14 out of 53 scenarios reaching the goal safely.

The results in Table 5.2 also reveal differences in safety across the scenario types. No collisions occurred in the first two scenarios types shown in Table 5.2. In contrast, for the last type of scenario with two cars and two pedestrians accounted for all 15 collisions, corresponding to a (28.3%) rate for this scenario type. Furthermore, 24 (45.3%) of the scenarios of this type remained safe but did not reach the goal position. Additionally, 48 (76.2%) and 18 (64.3%) of the other types of scenarios also remained safe but did not reach the goal position. Overall, even though most scenarios resulted in safe trajectories, successful goal completion is limited to (27.1%) out of all evaluated scenarios.

Scenario Type	Evaluated	Safe & Goal Not Reached	Nearest Actor <6m	Nearest Actor 6-20m	Nearest Actor 20-50m	Nearest Actor 50-100m	Nearest Actor >100m
Scenario with actors car, cyclist, and pedestrian	63	48 (76.2%)	5 (10.4%)	4 (8.3%)	4 (8.3%)	3 (6.3%)	32 (66.7%)
Scenario with actors car and cyclist	28	18 (64.3%)	2 (11.1%)	1 (5.6%)	2 (11.1%)	2 (11.1%)	11 (61.1%)
Scenario with actors 2 cars and 2 pedestrians	53	24 (45.3%)	4 (16.7%)	3 (12.5%)	3 (12.5%)	0 (0.0%)	14 (58.3%)
ALL	144	90 (62.5%)	11 (12.2%)	8 (8.9%)	9 (10.0%)	5 (5.6%)	57 (63.3%)

Table 5.3: The end-to-end planner’s distribution of minimum distances between the ego vehicle trajectory and the ground truth positions of the surrounding actors for safe scenarios that did not reach the goal position.

Looking back at Table 5.1, (62.5%) of the scenarios from the end-to-end-planner resulted in safe trajectories that did not reach the goal position. This raises the question of whether the end-to-end planner’s higher proportion of safe outcomes reflects an ability to actively plan around other actors, or whether these scenarios simply contained less interaction because the planner chose an ego vehicle trajectory far away from the intersection.

To investigate this, the minimum distance between the ego vehicle’s planned trajectory and the ground truth positions of the surrounding actors was examined for these 90 out of 144 evaluated scenarios. As shown in Table 5.3, 57 scenarios (63.3%) had a minimum distance greater than 100 meters from the nearest other actor. An additional five scenarios (5.6%) had a minimum distance between 50 and 100 meters. Consequently, 62 (68.9%) of all **Safe & Goal Not Reached** scenarios had a minimum distance of greater than 50 meters to any other actor throughout the scenario.

These results indicate that, in a substantial proportion of these 90 safe but goal not reached scenarios, the ego vehicle remained relatively far from other actors. The planner generated an ego vehicle trajectory that was far from both the intersection and the goal. Thus, the absence of collisions in these scenarios does not necessarily demonstrate that the planner successfully navigated around interacting actors. The remaining 28 (31.1%) scenarios had a minimum distance of 50 meters or less from another actor at some point during the scenario. Although these scenarios provide more relevant cases for investigated the end-to-end planner’s behavior in the presence of other actors, a minimum distance of 20 to 50 meters still represents a relatively large spatial separation, and therefore does not necessarily imply that the planner had to actively plan around a nearby actor.

5.4 Neuro-Symbolic Planner

This section presents the results of the neuro-symbolic planner in more detail. The analysis builds on the results presented in Table 5.1, following the same structure as

the analysis of the end-to-end planner. The scenario types are the same as those used for the end-to-end planner in Table 5.2, allowing for a direct comparison between the two planners. Moreover, the results from Table 5.3 are compared with the neuro-symbolic planner result for the **Safe & Goal Not Reached** category in Table 5.1. This comparison provides further insight into the safety performance of the two planners.

Scenario Type	Evaluated	Safe & Reached Goal	Safe & Goal Not Reached	Collisions
Scenario with actors car, cyclist, and pedestrian	63	37 (58.7%)	7 (11.1%)	19 (30.2%)
Scenario with actors car and cyclist	28	19 (67.9%)	3 (10.7%)	6 (21.4%)
Scenario with actors 2 cars and 2 pedestrians	53	29 (54.7%)	2 (3.8%)	22 (41.5%)
ALL	144	85 (59.0%)	12 (8.3%)	47 (32.6%)

Table 5.4: Neuro-symbolic planner result across different scenario types, showing the number and its percentage of evaluated scenarios that safely reached the goal, remained safe but did not reach the goal, or resulted in a collision.

Table 5.4 presents the results of the neuro-symbolic planner across the different types of scenarios. In total, the same 144 scenarios used for the end-to-end planner were evaluated, of which 85 (59.0%) scenarios were classified as **Safe & Goal Reached** and 12 (8.3%) were classified as **Safe & Goal Not Reached**, while the remaining 47 (32.6%) scenarios resulted in collisions. Overall, 97 (67.4%) of the evaluated scenarios resulted in a safe trajectory for the ego vehicle.

Compared with the end-to-end planner, which achieved a safe goal completion rate of (27.1%) as seen in Table 5.2, the neuro-symbolic planner has a much higher rate of safe trajectories that reached the goal, at (59.0%). However, this improvement in safe goal completion was followed by a higher collision rate. While the end-to-end planner resulted in a collision rate of (10.4%), the neuro-symbolic planner had a collision rate of (32.6%). These results highlight a difference in the two approaches, the neuro-symbolic planner reached the goal safely more frequently, but also resulted in a larger proportion of collisions.

These results also vary across the different scenario types. The highest safe goal completion rate (67.9%) was observed in scenarios with a car and a cyclist, followed by scenarios involving a car, a cyclist, and a pedestrian at (58.7%). Lastly, the scenarios with two cars and two pedestrians had both the lowest safe goal completion rate of (54.7%) and the highest collision rate of (41.5%).

The **Safe & Goal Not Reached** category accounted for only (8.3%) of all evaluated scenarios. The type of scenario that had the highest rate was the car, cyclist, and

pedestrian scenarios, while the lowest at a (3.8%) rate was the scenario types with two cars and two pedestrians. Overall, these results do not indicate a consistent relationship between the number of actors. Performance is varied across the scenarios types.

Scenario Type	Evaluated	Safe & Goal Not Reached	Nearest Actor <6m	Nearest Actor 6-20m	Nearest Actor 20-50m	Nearest Actor 50-100m	Nearest Actor >100m
Scenario with actors car, cyclist, and pedestrian	63	7 (11.1%)	4 (57.1%)	2 (28.6%)	1 (14.3%)	0 (0.0%)	0 (0.0%)
Scenario with actors car and cyclist	28	3 (10.7%)	0 (0.0%)	2 (66.7%)	0 (0.0%)	1 (33.3%)	0 (0.0%)
Scenario with actors 2 cars and 2 pedestrians	53	2 (3.8%)	0 (0.0%)	1 (50.0%)	1 (50.0%)	0 (0.0%)	0 (0.0%)
ALL	144	12 (8.3%)	4 (2.8%)	5 (3.5%)	2 (1.4%)	1 (0.6%)	0 (0.0%)

Table 5.5: The neuro-symbolic planner’s distribution of minimum distances between the ego vehicle trajectory and the ground truth positions of the surrounding actors for safe scenarios that did not reach the goal position.

To further examine the **Safe & Goal Not Reached** category, Table 5.5 presents the distribution of minimum distances between the ego vehicle and the ground truth positions of the surrounding actors for the 12 (8.3%) scenarios in this category. Of these 4 (2.8%) had a minimum distance of less than 6 meters, 5 (3.5%) had a minimum distance between 6 and 20 meters, 2 (1.4%) had a minimum distance between 20 and 50 meters, and only 1 (0.6%) had a minimum distance between 50 and 100 meters. None of the scenarios had a minimum distance greater than 100 meters. Thus, 11 out of the 12 scenarios (91.7%) had a minimum distance of 50 meters or less from another actor at some point along the planned trajectory.

A comparison with the end-to-end planner reveals a difference in the observed distance distributions for scenarios within the category **Safe & Goal Not Reached**. For the end-to-end planner 28 (31.1%) out of 90 scenarios in this category, had a minimum distance of 50 meters or less from another actor at some point during the scenario. Whereas, for the neuro-symbolic planner, 11 out of 12 (91.7%) scenarios had a nearest actor within 50 meters at some point along the planned trajectory. Conversely, 62 (68.9%) end-to-end planner scenarios had a minimum distance greater than 50 meters, compared with 1 (8.3%) neuro-symbolic planner scenario.

These results show that the generated trajectories by the planners differed in their spatial proximity to the surrounding actors when the goal was not reached. For the end-to-end planner, a large proportion of the generated trajectories remained relatively far from the other actors, whereas the neuro-symbolic planner predominantly produced trajectories in which the ego vehicle remained in closer proximity to the other actors. However,, proximity alone does not establish whether a planner actively responded to other actors positions or performed collision avoidance. These findings therefore suggest differences in the trajectory behavior of the two approaches, while

5. Results

further analysis would be needed to determine the extent to which these differences reflect active interaction with the surrounding actors.

6

Conclusion

The primary objective of this thesis was to assess whether a neuro-symbolic planner improves safety compared with a standard end-to-end deep-learning-based planner. This objective was motivated by the challenge of developing autonomous driving systems that can operate safely and reliably in complex traffic scenarios. To address this challenge, a neuro-symbolic approach was developed in which an LLM-based trajectory prediction module was combined with a symbolic A* planner. The neuro-symbolic approach was then evaluated against an end-to-end deep-learning-based planner in simulated intersection scenarios using quantitative safety and goal-reaching metrics.

The quantitative experimental results show that the neuro-symbolic planner performed best in terms of reaching the goal position safely. However, when comparing the overall safety performance of the two planners, the results present a more nuanced representation. The neuro-symbolic planner achieved a higher safe goal completion rate of (59.0%), compared with (27.1%) for the end-to-end planner, but it also had a higher collision rate of (32.6%) compared with (10.4%).

The distance analysis provides further insight into the trajectories classified as safe but not goal reaching. Although these trajectories were classified as safe, because no collision occurred, the distance analysis shows that the ego vehicle remained relatively far from the other actors in many of the scenarios where the goal was not reached. This suggests that, in these cases, collisions-free outcomes may have been achieved without substantial proximity to other actors. In contrast, the majority of scenarios in which the neuro-symbolic planner produced a safe trajectory without reaching the goal had a minimum distance of less than or equal to 50 meters from the other nearest actor. This indicates that the neuro-symbolic planner was generally able to produce a trajectory that both maintained safety and remained relevant to the scenario and the goal. Meanwhile, the end-to-end planner frequently produced collision-free trajectories far away from the nearest actor, suggesting limitations in its ability to produce trajectories that reflect the intended navigation of the scenario.

The LLM actor trajectory prediction results provide further context for the safety and goal-reaching performance of the neuro-symbolic planner. Although the LLM actor trajectory predictions were, for the most part, very accurate for the first two prediction steps, the fourth and fifth prediction steps often deviated substantially from the ground-truth trajectories. Since these predictions were used as input to the A* planner, inaccuracies in later prediction steps may have affected the resulting

planning trajectories. This highlights that the success of the neuro-symbolic approach is dependent on the accuracy of the LLM-based actor trajectory predictions.

The main takeaway from this thesis is that the neuro-symbolic approach significantly outperformed the end-to-end planner in terms of safely reaching the goal position. However, the neuro-symbolic planner also exhibited a higher collision rate. Therefore, the finding that the neuro-symbolic approach improves safety in general must be considered alongside the limitations of the LLM-based actor trajectory predictions and the observed differences in how the two planners achieved safety.

There are three main areas for future work that are particularly relevant based on the findings of this thesis. First, the number of actor trajectory prediction steps generated by the LLM could be reduced to investigate whether a shorter prediction horizon is associated with fewer collisions. The results showed that the fourth and fifth prediction steps were significantly less accurate than the earlier prediction steps. A shorter prediction horizon of two prediction steps, for which the LLM predictions were most accurate, could potentially improve the accuracy and reliability of the actor trajectory predictions used by the A* planner in the neuro-symbolic approach.

Second, the LLM used for actor trajectory prediction could be trained on a larger and more diverse dataset. Including more non-linear movements and a wider range of actor behaviors would expose the model to more complex trajectory patterns and could improve the model's ability to generalize to scenarios not represented in the current dataset. This could potentially lead to more accurate actor trajectory predictions and, consequently, improve the performance of the neuro-symbolic planner.

Lastly, an additional evaluation metric could be introduced to investigate the relationship between the accuracy of the LLM-based actor trajectories and the resulting number of collision scenarios. This would present further insight into whether inaccuracies in the actor trajectory predictions contribute to collisions and to what extent.

Bibliography

- [1] Z. Zhou, H. Huang, B. Li, S. Zhao, Y. Mu, and J. Wang, “Safedrive: Knowledge- and data-driven risk-sensitive decision-making for autonomous vehicles with large language models,” *Accident Analysis Prevention*, vol. 224, p. 108 299, 2026, ISSN: 0001-4575. DOI: <https://doi.org/10.1016/j.aap.2025.108299>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0001457525003872>.
- [2] M. Ganesan, S. Kandhasamy, B. Chokkalingam, and L. Mihet-Popa, “A comprehensive review on deep learning-based motion planning and end-to-end learning for self-driving vehicle,” *IEEE Access*, vol. 12, pp. 66 031–66 067, 2024. DOI: 10.1109/ACCESS.2024.3394869.
- [3] B. B. Elallid, N. Benamar, A. S. Hafid, T. Rachidi, and N. Mrani, “A comprehensive survey on the application of deep and reinforcement learning approaches in autonomous driving,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 7366–7390, 2022, ISSN: 1319-1578. DOI: <https://doi.org/10.1016/j.jksuci.2022.03.013>. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1319157822000970>.
- [4] S. Teng et al., “Motion planning for autonomous driving: The state of the art and future perspectives,” *IEEE Transactions on Intelligent Vehicles*, vol. 8, no. 6, pp. 3692–3711, Jun. 2023, ISSN: 2379-8858. DOI: 10.1109/tiv.2023.3274536. [Online]. Available: <http://dx.doi.org/10.1109/TIV.2023.3274536>.
- [5] Y. Xu et al., *Trajectory prediction meets large language models: A survey*, 2025. arXiv: 2506.03408 [cs.CL]. [Online]. Available: <https://arxiv.org/abs/2506.03408>.
- [6] D. Pesjak and J. Žabkar, “Robot planning via llm proposals and symbolic verification,” *Machine Learning and Knowledge Extraction*, vol. 8, no. 1, 2026, ISSN: 2504-4990. DOI: 10.3390/make8010022. [Online]. Available: <https://www.mdpi.com/2504-4990/8/1/22>.
- [7] M. Bojarski et al., “End to end learning for self-driving cars,” *arXiv preprint arXiv:1604.07316*, 2016. arXiv: 1604.07316 [cs.CV]. [Online]. Available: <https://arxiv.org/abs/1604.07316>.
- [8] Y. Lin, S. Illing, and M. Althoff, *Sandra: Safe large-language-model-based decision making for automated vehicles using reachability analysis*, 2025. arXiv: 2510.06717 [cs.RO]. [Online]. Available: <https://arxiv.org/abs/2510.06717>.

- [9] P. Hitzler and M. K. Sarker, Eds., *Neuro-Symbolic Artificial Intelligence: The State of the Art* (Frontiers in Artificial Intelligence and Applications). Amsterdam: IOS Press, 2022, vol. 342, ISBN: 978-1-64368-244-0.
- [10] W. X. Zhao et al., *A survey of large language models*, Mar. 2023. [Online]. Available: <https://arxiv.org/abs/2303.18223>.
- [11] Z. Wan et al., *Efficient large Language Models: a survey*, Dec. 2023. [Online]. Available: <https://arxiv.org/abs/2312.03863>.
- [12] Y. Zhang et al., *Siren’s Song in the AI Ocean: A survey on hallucination in large Language models*, Sep. 2023. [Online]. Available: <https://arxiv.org/abs/2309.01219>.
- [13] X. Han et al., “Parameter-efficient fine-tuning for large models: A comprehensive survey,” *arXiv preprint arXiv:2403.14608*, 2024.
- [14] E. J. Hu et al., “Lora: Low-rank adaptation of large language models,” *arXiv preprint arXiv:2106.09685*, 2021.
- [15] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “Qlora: Efficient finetuning of quantized llms,” *arXiv preprint arXiv:2305.14314*, 2023. DOI: 10.48550/arXiv.2305.14314. [Online]. Available: <https://arxiv.org/abs/2305.14314>.
- [16] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016. DOI: 10.1109/TIV.2016.2579553.
- [17] L. Chu et al., “Intelligent vehicle path planning based on optimized a* algorithm,” *Sensors*, vol. 24, no. 10, p. 3149, May 2024. DOI: 10.3390/s24103149. [Online]. Available: <https://doi.org/10.3390/s24103149>.
- [18] M. K. Ramasamy, *Why jsonl is a game-changer for llm development: A developer’s perspective*, <https://blog.gopenai.com/why-jsonl-is-a-game-changer-for-llm-development-a-developers-perspective-9a72330d7945>, [Online] Accessed May 15, 2026, 2025.