Research Internship
Report

Synthetic Data Generation for Testing and Enhancing the Robustness of Decision-Making Models in Autonomous Vehicles

Submitted by

MARTIN Sébastien
MVA Student / R&D Intern
ENS Paris-Saclay

Under the guidance of

CHARRAUT Valentin
Artificial Intelligence Engineer

MVA : Mathématiques, Vision et Apprentissage
ENS Paris-Saclay
4 Avenue des Sciences, Gif-sur-Yvette

April to October 2025

Abstract

Autonomous driving systems need to handle not only routine traffic but also rare, safety-critical events. Public datasets contain mostly ordinary scenarios, which limits the ability of models to generalize and creates a gap between training and deployment. This internship explores methods to generate synthetic scenarios with adjustable difficulty to help bridge that gap.

The work extends SMART, a next-token trajectory prediction model, with a conditioning mechanism based on agent-level metrics. Quantities such as acceleration and distance to lane center are encoded into a latent space using a variational autoencoder. The resulting latent vectors are injected into SMART to influence the generation of trajectories, with the aim of producing scenarios that cover a broader range of difficulty levels.

Baselines such as the Intelligent Driver Model were implemented for comparison. Evaluation followed the Waymo Open Sim Agents Challenge protocol, and used metrics that capture realism (e.g. adherence to road rules and motion statistics) and controllability (e.g. collision rate). Initial experiments confirm that difficulty embeddings can be learned and used in generation, though full conditioning of SMART is still in progress.

Future work will extend these experiments to larger scales, explore alternative generative models, and refine evaluation criteria to better capture the trade-off between realism and controllability. The overall objective is to build a practical framework for generating scenarios of varied difficulty, providing a more reliable basis for testing and improving autonomous driving systems.

Contents

Chapter 1

Introduction

1.1 Overview

The development of autonomous driving

The development of autonomous driving technologies is driven by a clear societal need. Road accidents remain one of the leading causes of fatalities worldwide, with human error being responsible for the vast majority of cases [57]. Beyond safety, autonomous vehicles hold the promise of improving traffic efficiency [38], reducing environmental impact through smoother driving patterns [20], and extending mobility to populations with limited access, such as the elderly or disabled [26]. These motivations have established autonomous driving as one of the most significant technological frontiers of the past decade.

The arrival of large-scale industrial initiatives has accelerated this transformation. Tesla has pioneered the deployment of semi-autonomous functionalities through its Autopilot system [47], while Waymo, emerging from Google's self-driving car project, has positioned itself as a leader in fully driverless technology [52]. Other companies such as Cruise, Baidu, and Aurora have also entered the field, contributing to an increasingly competitive landscape. These efforts have turned autonomous driving from an academic aspiration into a concrete industrial race, with billions of dollars invested and fleets deployed across the world.

Parallel to these industrial efforts, research around autonomous driving has intensified at an unprecedented pace. Major conferences in machine learning and robotics, such as NeurIPS [37], CVPR [3], and ICRA [4], now dedicate specific workshops and challenges to autonomous driving. The release of large-scale benchmarks such as the Waymo Open Dataset [43], nuScenes [8], and Argoverse [10] has further structured the field, allowing standardized evaluation and

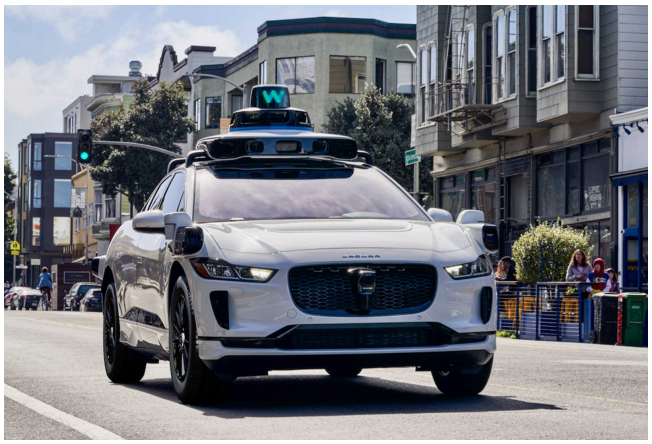


Figure 1.1: Waymo autonomous taxi

fostering rapid progress. These initiatives illustrate the convergence of academia and industry in addressing the complex challenges of perception, prediction, and planning in real-world driving environments.

Valeo and Its Role in Autonomous Driving

Valeo [5] is a major international automotive supplier, historically recognized for its contributions to vehicle electrification, comfort, and safety systems. In recent years, the company has expanded its portfolio by investing in advanced driver assistance systems (ADAS) and autonomous mobility solutions, positioning itself as a significant actor in the global transition toward automated driving [49].

One of Valeo's strengths lies in its expertise in automotive sensors. The company was among the first to industrialize automotive-grade LiDAR (Light Detection and Ranging) technology, which provides precise 3D perception of the vehicle's surroundings. LiDAR has become a key enabler for higher levels of autonomy, complementing cameras and radar sensors. Building on this technological base, Valeo has recently started to develop and propose Reinforcement Learning based autonomous driving and self-parking solutions. These systems aim to support both partial autonomy in everyday traffic as well as fully automated maneuvers in constrained environments such as parking lots.

The company's ambition is not to compete head-to-head with tech giants like Tesla or Waymo in the deployment of fully driverless cars, but rather to leverage its industrial expertise and partnerships with automakers to deliver scalable, safe, and cost-efficient solutions. In practice, this means developing decision-making and perception systems that can be embedded directly into mass-produced vehicles, focusing on real-world usability and compliance with safety regulations.

Valeo's recent move into autonomous driving reflects a broader trend in the automotive industry, where traditional suppliers evolve into key innovation drivers for ADAS and autonomy. By entering this space, Valeo joins the ongoing global effort to make intelligent driving assistance more reliable and to gradually push toward higher levels of vehicle automation.

1.2 Main challenges

Paradigms of Decision Models in Autonomous Driving

At the core of autonomous driving lies the decision model, the component responsible for determining how the vehicle should behave in traffic. Broadly speaking, there are two major paradigms in designing these models: end-to-end learning and mid-to-end learning.

- In end-to-end learning, a model directly maps raw sensor data (such as camera images, LiDAR point clouds, or radar signals (see figure 1.2)) to driving actions like steering angle, acceleration, or braking. This approach has the advantage of simplicity, as it avoids hand-designed intermediate stages. However, it often suffers from limited interpretability and can be difficult to train robustly, since the model must simultaneously learn perception, prediction, and decision-making.

- On the other hand, the mid-to-end approach decomposes the driving problem into intermediate representations before making a final decision. For instance, sensor data are first transformed into a bird's-eye-view (BEV) map (see figure 1.3) containing the environment and trajectories of surrounding agents, before the decision module outputs the ego vehicle's next actions. This additional structure often improves interpretability, safety, and sample efficiency, at the cost of increased system complexity.

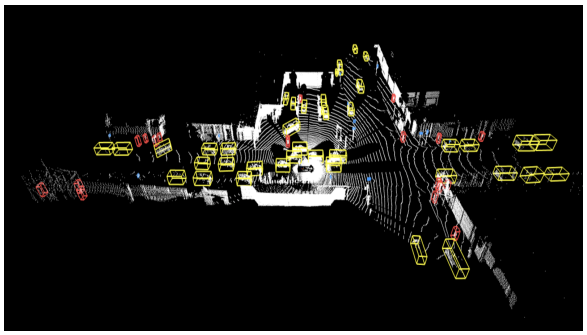


Figure 1.2: Waymo car perception

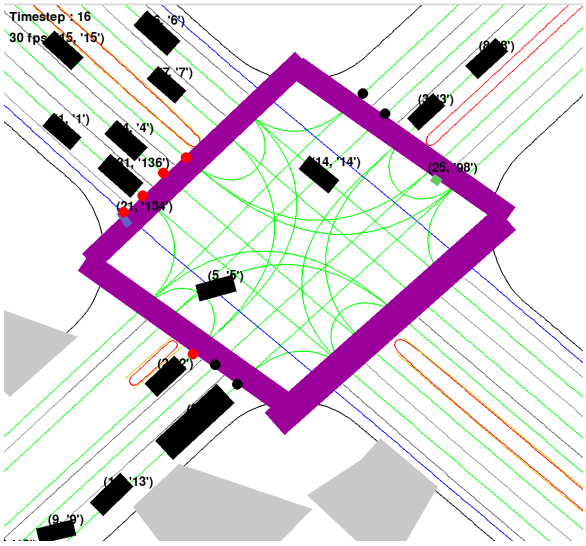


Figure 1.3: A bird-eye-view scenario example

Mid-to-End Decision Approaches

Within the mid-to-end paradigm, several types of approaches have been explored.

- Rule-based methods rely on pre-defined models of driver behavior. A well-known example is the Intelligent Driver Model (IDM) [48], which describes car-following by balancing safety distance and desired speed. Such methods are interpretable and computationally efficient, but often fail to generalize to the full complexity of real-world traffic.
- Reinforcement learning (RL) [46] frames driving as a sequential decision-making problem, where an agent learns by interacting with an environment and receiving rewards for safe and efficient behavior. RL has the potential to produce highly adaptive strategies, but requires careful reward design and large amounts of training data.
- Imitation learning (IL) [1] directly learns driving policies from demonstrations provided by human experts. By observing large amounts of real-world driving data, IL enables models to reproduce human-like driving patterns without explicit reward design. However, it can inherit human errors and struggles in situations that were not represented in the training data.

1.3 Problematics of the domain

The Role of Scenarios

In the context of autonomous driving, a scenario refers to a structured traffic situation involving multiple agents (such as cars, pedestrians, or cyclists) interacting in a specific road layout. Examples include merging onto a highway, negotiating a four-way stop, or overtaking on a two-lane road.

For imitation learning, scenarios are essential since the model learns directly from recorded expert demonstrations. Without realistic and diverse scenarios, imitation learning risks producing policies that fail in uncommon but critical situations.

For reinforcement learning, scenarios are equally important, although their role is slightly different. Training in RL often involves either log-replay scenarios, where recorded traffic

data is replayed in simulation but without reactivity from other agents, or synthetic scenarios, generated from real data to populate the environment. These approaches provide realism but may lack adaptability, since surrounding vehicles do not always react to the ego vehicle's decisions. While there are exceptions — such as rule-based reactive traffic models or multi-agent RL, where other vehicles also learn behaviors — the majority of RL training still depends on replayed or data-driven synthetic scenarios.

The Need for Harder Scenarios

One persistent issue in publicly available driving datasets is that they tend to be dominated by routine, uneventful traffic. Truly complex or dangerous situations — sudden braking, near-collisions, or aggressive cut-ins — are extremely rare. This imbalance creates a distribution shift: models trained primarily on "easy" data may struggle when encountering rare but critical events in the real world.

To bridge this gap, researchers are increasingly turning toward the creation of synthetic, data-driven scenarios. The idea is to preserve the human-like behavior observed in real data, while also enhancing the difficulty of situations to deliberately stress-test driving models. For imitation learning, this means enriching the dataset with challenging examples; for reinforcement learning, it means designing environments where the ego agent faces rare but important corner cases. In both cases, harder and more diverse scenarios are essential to developing decision models that remain robust under the unpredictability of real-world driving.

1.4 Internship Objectives

Objective of the Internship

The main objective of this internship is to design and implement methods for scenario generation in the field of autonomous driving, with a particular focus on the creation of difficult and controllable scenarios. Scenario generation plays a critical role in bridging the gap between real-world data, which often lacks complex or dangerous cases, and the need to expose learning algorithms to challenging situations. By simulating diverse and realistic interactions between vehicles, pedestrians, and traffic rules, scenario generation allows us to build environments that go beyond what can be safely or efficiently collected on the road.

Controllability of Scenarios

A central ambition of the internship is to develop models that not only generate scenarios, but also allow their difficulty to be adapted. This means designing frameworks where factors such as traffic density, the behavior of surrounding agents, or the complexity of the road layout can be tuned to create situations ranging from ordinary to highly critical. Such controllable generation is key to ensuring that driving policies are not only trained on routine scenarios but are also confronted with rare and safety-critical cases that truly test their robustness.

Validation of Generated Scenarios

The work will also involve validating that the generated scenarios are both realistic and effective for training and testing driving models, even if this is not the primary focus. Demonstrating that controllable difficulty can be introduced without losing realism is an important step to show the usefulness of the proposed methods.

1.5 Contribution

The subject of scenario generation for autonomous driving is broad and ambitious, and it quickly became clear that the scope was too large to be fully covered within the timeframe of an internship. On top of that, the writing of this report had to begin around two-thirds of the way through the internship, meaning that not all of the planned work could be completed or included here. The results presented therefore concentrate on a few key directions, while the Future Work section at the end of this manuscript will outline possible follow-ups that go beyond what could be achieved during this project.

Literature Review

The first step of the internship was a comprehensive review of the field, covering existing datasets, driving simulators, approaches for policy creation, methods for scenario generation, and recent works on varied-difficulty scenarios. This study, which will be presented in detail in the Related Works section, was essential for grounding the contributions in the state of the art and for identifying gaps where new approaches could be explored.

Main Contribution: Controlling Scenario Difficulty

The main contribution of the internship is the proposal of a new approach to control scenario difficulty. Instead of relying solely on handcrafted scenario parameters, the idea is to use metrics directly measurable on each agent — such as speed, acceleration, or trajectory deviation — and embed them into a regularized latent space. This embedding yields a latent vector that characterizes the behavior of each agent from a difficulty perspective. By conditioning existing scenario generation models on these latent vectors, we can effectively guide the difficulty level of each agent in the scene.

This design enables two powerful features:

- the ability to control and shift the difficulty of scenarios in a systematic way,
- and the possibility to sample directly in the latent space to create new scenarios where each agent is assigned a distinct, random difficulty profile.

Through this approach, scenario generation becomes not only more controllable but also more flexible, offering a practical way to enrich datasets with diverse, agent-level difficulty variations that go beyond traditional scenario design.

Re-implementation of Baselines

The first part of the work consisted in re-implementing existing methods that would be useful for the rest of the internship. SMART, winner of the last Waymo challenge on simulation of agents and pioneer in a new way of scenario generation, was first re-implemented. At the same time, CATK, one of the successors of SMART and among the leaders of the same challenge at the time of writing, was also reproduced. These models serve both as baselines for comparison and as the very base of the proposed method, since the SMART architecture is the one that will be conditioned in order to create the shifting difficulty model. In addition, another important baseline was implemented: the Intelligent Driver Model, a rule-based deterministic agent model that allows comparison of the proposed approach with a classical method from a different paradigm.

Embedding Difficulty Metrics

The second part of the work focused on the core of the experiment, starting with the choice of metrics used to condition the agents and the justification for their use. These metrics were embedded into a compact representation of difficulty, providing the latent vectors used to guide generation. Evaluation metrics were then defined to compare approaches on both realism and controllability. The conditioning of SMART with these embeddings has been implemented, but full experimentation is still in progress at the time of writing.

Experimental Progress and Limitations

The experiments mark the transition from methodology to validation. The baseline models and embedding framework are operational, and preliminary analyses confirm that the latent space is well structured. However, the conditioning runs and quantitative evaluation remain incomplete due to the internship timeline. The current stage can therefore be seen as an intermediate milestone: the tools are ready, and the first steps of validation are set up, but conclusive results are not yet available.

Future Work

The Future Work section will extend this trajectory by addressing unfinished experiments and broadening the scope of evaluation. The aim is to consolidate the proof of concept into a complete framework for controllable scenario generation. This includes finishing the conditioning experiments, comparing against additional baselines, and testing more diverse difficulty metrics. Beyond immediate follow-ups, the prospects also include exploring alternative generative architectures and integrating scenario evaluation directly with autonomous driving policies, in order to assess robustness in a realistic closed loop.

Chapter 2

Related works

In this chapter, we review the different areas of research that are relevant to the generation of shifted-difficulty scenarios in autonomous driving. Our goal is to situate our contribution within the broader context of scenario generation, dataset design, evaluation protocols, and conditioning strategies.

We begin with a brief overview of methods for *policy generation*, which form the algorithmic foundation of most scenario-generation pipelines. These methods, and the assumptions they rely on, are central to understanding how agents can be modeled and how their interactions emerge.

Next, we introduce the main autonomous driving datasets that support motion generation research. We highlight their contents, properties, and formats, emphasizing how these factors influence the design and applicability of generative models. Among these datasets, we focus more specifically on the **Waymo Open Motion Dataset** [54], which is the dataset used throughout this work. We discuss in detail its structure, the information it provides, and the preprocessing required to make it suitable for hands-on experimentation and follow-ups.

This naturally connects to the **Waymo Open Challenges**, where we outline their goals, inputs, and outputs. Understanding these challenges is crucial, as they provide the evaluation benchmarks and community standards that guide current progress in scenario generation.

Building on these foundations, we then discuss what constitutes a *good* scenario in autonomous driving research: the notions of realism, diversity, safety, and controllability, as well as how these criteria can be measured in practice. This leads us to a review of existing approaches for **scenario generation**, with a particular emphasis on methods that attempt to modulate or control the *difficulty* of generated trajectories. We analyze their commonalities, strengths, and limitations, in order to clarify how our proposed approach both aligns with and extends beyond the current state of the art.

2.1 Driving Policies

The 3 paradigm approached for designing driving policy are rule-based, imitation learning and reinforcement learning.

Rule-based

Rule-based driving policies aim to capture the complexity of driving through predefined rules. A widely used example is the Intelligent Driver Model (IDM) [48], which computes vehicle speed by balancing safety considerations with the desired velocity. However, IDM is not flawless [6]; it can encounter failure cases that lead to divergence. Its directional control is also

rather simplistic, as it primarily relies on lane-center following. Despite these limitations, research on IDM remains an active area [68], driven by its strong potential for interpretability and computational efficiency. Our implementation of IDM will be detailed later in the **Experiments and Results** section 7.1.2. More recently, rule-based approaches have been extended with predictive elements. The Predictive Driver Model (PDM) [17] illustrates this direction: instead of simply staying in the middle of the lane, it generates and evaluates multiple candidate trajectories to choose a safe and efficient path. While still rooted in hand-crafted priors, such models demonstrate that interpretability and efficiency remain attractive, even as research increasingly shifts toward learning-based policies.

Imitation Learning

Imitation Learning (IL) has been a central paradigm for autonomous driving, leveraging expert demonstrations to directly transfer human-like behavior into learned policies. *PLUTO* [13] exemplifies the mid-to-end IL approach, operating on post-perception representations. Its strengths lie in an architecture explicitly modeling both longitudinal and lateral behaviors, a suite of auxiliary losses to enforce safety, and data augmentations to mitigate distribution shift. PLUTO demonstrated state-of-the-art performance on the nuPlan benchmark, surpassing even strong rule-based baselines such as PDM. Another line of IL research focuses on end-to-end policies, directly learning from sensor inputs. *Roach* [66] advances this paradigm by employing a reinforcement learning (RL) coach. The RL-based expert provides dense on-policy supervision, alleviating covariate shift and offering richer training signals than traditional human or rule-based experts. Similarly, *GAIL* [15] introduces an adversarial framework where the discriminator distinguishes expert from agent trajectories. This formulation allows learning robust end-to-end driving policies without requiring explicit reward design, though training stability remains a challenge. Finally, *Imitation is not Enough* [35] highlights the limitations of IL alone in rare or safety-critical scenarios. It proposes combining IL with RL through a simple reward function, achieving significantly higher robustness on challenging urban scenes while retaining the naturalistic behavior of imitation-trained policies.

Reinforcement Learning

Reinforcement Learning (RL) offers a compelling alternative to imitation learning by optimizing policies through closed-loop interaction. In the mid-to-end paradigm, CaRL [28] stands out with its remarkably simple reward formulation—route completion penalized for infractions—enabling massive-batch PPO training over hundreds of millions of samples and achieving superior performance over baselines like Roach on both CARLA [19] and nuPlan [9]. As an illustrative example of end-to-end RL, Ramble [33] embraces raw sensor inputs—including RGB images and LiDAR point clouds—and encodes them into latent features via a transformer-based, model-based architecture. By learning environment dynamics, Ramble anticipates future states and manages complex, dynamic, high-interaction traffic situations. It attains state-of-the-art route completion rates on CARLA Leaderboard 2.0, showcasing that end-to-end RL can indeed compete with mid-to-end methods, particularly in dense, interactive scenarios. At the extreme end of simulation-based learning, GIGAFLOW [16] takes a different self-play RL route, training a generalist policy in a self-play multi-agent manner, entirely in simulation without real-world data. It achieves zero-shot transfer across CARLA [19], nuPlan [9], and Waymax [24], demonstrating that large-scale simulation alone can yield robust, generalizable driving behaviors, with millions of kilometers between incidents.

2.2 Datasets

The datasets

A wide range of large-scale datasets has been released in recent years to support research in perception, prediction, and decision-making for autonomous driving. They provide diverse sensor recordings, annotations, and maps that allow researchers to benchmark and compare methods under realistic conditions. Below, we briefly present the main datasets relevant to this work.

Waymo Open Dataset. The Waymo Open Dataset [44], released in 2019, is one of the largest resources available today. It was collected in several US cities with Waymo's self-driving fleet and includes both LiDAR and camera recordings. The dataset contains thousands of short driving segments covering a wide variety of traffic, weather, and lighting conditions. Each segment is annotated with 2D and 3D bounding boxes for vehicles, pedestrians, cyclists, and traffic signs. Thanks to its size and diversity, it has become a standard benchmark for detection, tracking, and segmentation tasks.

nuScenes. The nuScenes dataset [8], released the same year, was the first to provide a complete 360° sensor setup with cameras, LiDAR, radars, and GPS/IMU. It consists of 1,000 scenes recorded in Boston and Singapore, two cities chosen for their dense and complex traffic. All scenes are carefully annotated with 3D bounding boxes and additional attributes such as pedestrian actions or vehicle pose. NuScenes also provides detailed semantic maps of the driving areas, making it particularly valuable for research on 3D detection, sensor fusion, and trajectory forecasting.

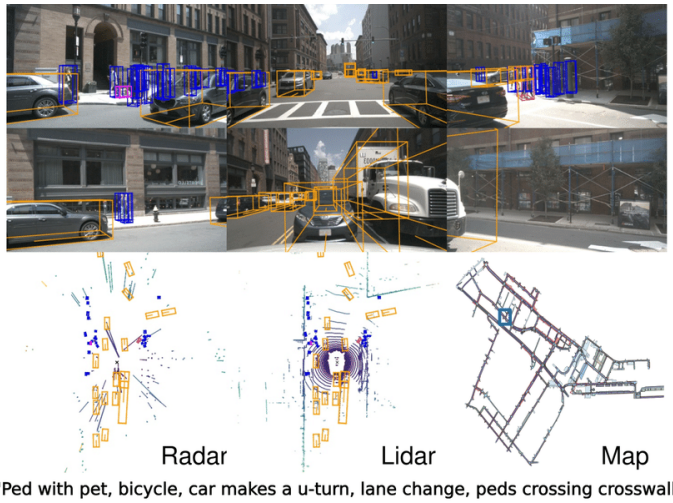


Figure 2.1: An example from the nuScenes dataset, combining multiple camera views, lidar and radar data, a semantic map, and a textual description.

Argoverse. The Argoverse dataset [10], introduced by Argo AI, emphasizes the link between perception and motion forecasting. Collected in Miami and Pittsburgh, it includes two parts: a 3D tracking set with annotated object sequences, and a motion forecasting set with over 300,000 scenarios containing future trajectories of surrounding agents. A key feature of Argoverse is its provision of high-definition maps with lane centerlines and intersection topology, which allow algorithms to use map context for more accurate prediction.

Waymo Open Motion Dataset. All experiments in this report are conducted on the Waymo Open Motion Dataset (WOMD) [54], which focuses specifically on motion prediction. Instead of raw sensor data, WOMD provides bird's-eye-view (BEV) scenes showing the motion of vehicles, cyclists, and pedestrians over time. It was released together with three challenges: the Sim Agents Challenge [56], the Motion Generation Challenge [55], and the Interaction Prediction Challenge [53]. These benchmarks have quickly become reference points for mid-to-end research in trajectory forecasting.

Because WOMD uses a complex format, in this work the dataset was converted into the *ScenarioMax* format [50], an extension of ScenarioNet [32]. ScenarioMax unifies multiple datasets (Waymo, nuScenes, nuPlan) into a common structure and can export simulator-ready inputs for Waymax [24], V-Max [11], and GPUDrive [30]. This conversion is particularly valuable for our experiments, as it allows compatibility with several simulation tools while enabling flexible scenario augmentation and controlled difficulty levels.

Waymo Open Motion Dataset Challenges

To foster progress in motion forecasting and simulation, three challenges have been introduced around the **Waymo Open Motion Dataset**: *Interaction Prediction* [53], *Sim Agents* [56], and *Scenario Generation* [55].

- The **Interaction Prediction Challenge** targets short-term forecasting of interactive behaviors. Given a brief history of motion on a map, the task is to jointly predict how two selected agents (vehicles, pedestrians, or cyclists) will move into the future. Performance is assessed by the accuracy and diversity of the predicted joint trajectories.
- The **Sim Agents Challenge** extends this to full-scene simulation. From one second of past history for all agents, participants must generate multiple plausible futures capturing realistic motions, interactions, and adherence to traffic rules. This setting emphasizes the creation of scene-consistent rollouts rather than isolated predictions.
- Finally, the **Scenario Generation Challenge** goes further by asking participants to create entire scenes from minimal input: only the ego agent's history, the map, and the number of agents to include. The goal is to generate both the initial states and the full trajectories of all agents in a way that resembles real-world driving.

Together, these challenges push research beyond single-agent prediction toward richer simulation and scenario generation, encouraging methods that can model complex, interactive traffic behavior. As a result, they have also stimulated the emergence of a wide variety of **generative approaches** for realistic trajectory and scenario modeling.

2.3 Evaluation of Scenario Generation models

Before diving into Simulation and Scenario Generation models, it is important to establish how the quality of such models can be measured. Two main characteristics are typically assessed: the difficulty of generated scenarios and their realism. An effective model is one that maintains a good balance between them.

Difficulty

While the notion of a "difficult" driving scenario might seem intuitive, quantifying this difficulty rigorously is challenging. Broadly speaking, two approaches are commonly adopted:

- **Metric-based evaluation** Difficulty can be estimated through predefined metrics such as **collision rate**, **minimum time-to-collision**, or **comfort indicators** (acceleration, jerk, angular speed). For instance, *AdvDiffuser* evaluates adversarial scenarios using collision frequency, impact severity, and maneuver comfort [60]. Likewise, *KING* reports the success rate of generating collisions as well as their severity [25], while *FREA* introduces feasibility-oriented measures such as the **Infeasible Ratio** (how often the ego vehicle has no safe option) to ensure generated challenges remain solvable [12]. These quantitative metrics are comparable across scenarios, but they often fail to capture subtler forms of difficulty such as nuanced right-of-way interactions.

- **Policy-based evaluation** Another perspective is to measure how a trained driving policy performs in the generated scenarios. Here, difficulty is tied to the observed performance degradation or safety violations of the policy under test. This approach directly reflects how scenarios affect downstream learning, but results vary significantly depending on which policy is chosen and how it was trained.

Realism

Realism, by contrast, is usually assessed through distributional comparisons. The idea is to check whether generated scenarios exhibit feature distributions (e.g., speeds, accelerations, headways) similar to those found in real-world traffic data. Works such as *SMART* and *VBD* evaluate realism by comparing trajectory distributions against logged data [59, 27], while *TrafficGamer* emphasizes adherence to maps and plausible multi-agent interactions [40]. *ChatScene* adds domain knowledge and measures both safety (collision rate) and functionality (rule compliance, smoothness) to validate the realism of its generated cases [64]. *CollisionGen* focuses specifically on accident realism by aligning generated crash distributions with real-world accidents [51].

The most systematic evaluations come from the **Waymo Open Motion Dataset Challenges**, which have established widely used benchmarks. The *Sim Agents Challenge* [56] requires generating multiple rollouts for all agents given one second of history. Realism is then assessed through three categories of metrics: (i) **agent motion**, measuring whether speed, acceleration, and yaw distributions match real traffic; (ii) **agent interactions**, checking collisions, distances between vehicles, and time-to-collision statistics; and (iii) **map adherence**, monitoring off-road frequency, distance to lane boundaries, and traffic-light violations. The *Scenario Generation Challenge* [55] extends this to the creation of entire scenes, where evaluation is based on the likelihood of logged data under the generated distribution, aggregated across motion, interaction, and map features. Both challenges summarize results into a composite "realism meta-metric," which balances these aspects and has become a standard for the community.

2.4 Scenario Generation Models

Research on scenario generation for autonomous driving has advanced along three complementary directions: producing **realistic traffic behavior**, developing **controllable/conditional models** that allow guided simulation, and designing **safety-critical generators** that expose autonomous systems to rare but important failures.

Realistic scenario generation.

The strongest results in realism come from *GPT-like next-token prediction. SMART* discretizes scenes and autoregressively predicts multi-agent motion, achieving state-of-the-art realism on

the Waymo Sim Agents benchmark and inspiring many leaderboard variants that refine tokenization, architecture, or training while keeping the same principle [59]. *Trajeglish* follows a similar idea by treating driving as a "language of motion," tokenizing trajectories with high precision and improving interaction/realism metrics [39]. Diffusion-based approaches also perform strongly: *VBD* (Versatile Behavior Diffusion) generates diverse, coherent multi-agent behaviors and supports inference-time editing [27]; *SceneDiffuser* introduces amortized diffusion and adds explicit conditioning (e.g., map or language prompts), reaching top closed-loop realism [29]; *Scenario Dreamer* combines latent diffusion for scene initialization with transformers for motion rollout to efficiently generate complete new scenes with high diversity [41]. Together, these works explain why **next-token transformers and diffusion** currently dominate realism benchmarks.

Controllable and conditional generation.

Beyond unconditional realism, recent work emphasizes **guidance and controllability**. *CtRL-Sim* conditions multi-agent policies on return signals from offline RL, so adjusting reward weights steers behaviors (e.g., cautious vs. aggressive) [42]. *RealGen* retrieves and recombines snippets from real logs to assemble scenarios that match user-specified conditions [18]. Natural-language control further broadens access: *ChatScene* translates text instructions into executable scenarios, leveraging knowledge retrieval to create diverse, complex cases [64]. Diffusion methods also integrate conditioning—*SceneDiffuser* accepts map/text prompts, while *VBD* supports controlled refinements at inference [29, 27]. These approaches enable targeted testing within a realistic generative framework.

Safety-critical and adversarial generation.

A complementary direction targets **rare, safety-critical situations**. *FREA* creates near-miss events while guaranteeing feasibility for the ego vehicle via a Largest Feasible Region criterion [12]. *TrafficGamer* employs game-theoretic oracles to elicit realistic yet risky multi-agent interactions [40]. *SelfPlay* adopts an asymmetric teacher–student setup that produces solvable-but-failing scenarios for curriculum-style robustness gains [63]. *KING* uses kinematic gradients to directly optimize background trajectories and induce ego failures [25]. Generative techniques have also been adapted to adversarial goals: *AdvDiffuser* steers diffusion sampling using adversarial objectives while preserving realism [60]; *ReGentS* and *CollisionGen* perturb or compose real trajectories under constraints to build scalable, controlled crash sets [62, 51].

Chapter 3

Problem

Aim and scope. The main goal of this work is to create a model-agnostic way to control the difficulty of generated driving scenarios. By "universal," we mean a conditioning interface that can be plugged into different types of generative models (e.g., token autoregressors, diffusion models) without changing how the generator itself works. As a concrete testbed, we instantiate and study this hypothesis on a **GPT-like next token prediction** model, which currently represents a strong foundation for multi-agent motion generation. We also specifically investigate difficulty conditioning *within the next-token prediction paradigm*, for which we are not aware of prior, explicit difficulty-control interfaces.

Background on Next-token prediction. Autonomous driving motion generation models aim to produce realistic, diverse, and safe trajectories for multiple interacting agents in a traffic scene. GPT-like next token predictors treat driving as a sequence modeling problem: trajectories and road vectors are tokenized into discrete symbols, and a decoder-only transformer predicts the next token autoregressively. Formally, the predictor models the conditional distribution

$$p_\theta(\tau^1, \ldots, \tau^{N_A} \mid \mathcal{C}), \tag{3.1}$$

where $\tau^i = (a_0^i, \ldots, a_T^i)$ is the tokenized trajectory of agent i, and $\mathcal{C}$ represents the context (map tokens, historical tokens, and multi-agent interactions).

Limitation we address. While **next token prediction** model effectively captures scene dynamics and generates plausible scenarios, it does not provide a direct mechanism to *control the difficulty level* of a scenario. In practice, one may want to shift a scenario toward "easier" or "harder" conditions—for example, by making agent behaviors more aggressive, risky, or constrained.

Problem statement. Given a context $\mathcal{C}$, we aim to generate trajectories $\{\tau^i\}_{i=1}^{N_A}$ *conditioned not only on* $\mathcal{C}$, but also on a controllable difficulty signal:

$$\textbf{Given } \mathcal{C}, \text{ generate } \{\tau^i\}_{i=1}^{N_A} \text{ conditioned on } \mathcal{C} \text{ and a difficulty code.} \tag{3.2}$$

Concretely, we posit a per-agent latent difficulty vector $z^i \in R^Z$ and require

$$p_\theta(\tau^1, \ldots, \tau^{N_A} \mid \mathcal{C}, z^1, \ldots, z^{N_A}), \tag{3.3}$$

where each z^i encodes the *difficulty level* associated with agent i. This conditioning enables us to shift the realism of the generated scene along a spectrum of difficulty, thereby enriching simulation diversity and controllability, and serves as a first step toward a *universal* difficulty-conditioning interface for scenario generation.

Chapter 4

Methodology

Our approach extends the **SMART** framework, which models autonomous driving scenarios as a multi-agent next-token prediction task. In this section, we first formalize the variables used in our setting, then recall the foundations of SMART, and finally introduce our difficulty-aware conditioning mechanism.

4.1 Notation and variables

We consider a driving scene composed of N_A agents and a discretized time horizon of length T.

- The state of agent i at time t is denoted $s_t^i \in R^d$ (e.g., position, velocity, heading).
- The full trajectory of agent i is $\tau^i = (s_0^i, s_1^i, \ldots, s_T^i)$.
- We define K difficulty metrics m_k, each mapping a state to a scalar, and compute the *difficulty matrix* of agent i:

$$D^i \in R^{T \times K}, \quad D^i_{t,k} = m_k(s_t^i). \tag{4.1}$$

  This matrix encodes, over time, the physical or safety-related difficulty of the trajectory.

- A Variational Autoencoder (VAE) encodes each difficulty matrix into a latent vector $z^i \in R^Z$, where Z denotes the latent space dimension.
- The road network of the scene is tokenized into a set of N_R tokens $R = \{r_1, \ldots, r_{N_R}\}$, with $r_j \in \mathcal{V}_R$ where $\mathcal{V}_R$ is the road vocabulary.
- Similarly, each trajectory τ^i is tokenized into motion tokens $(a_0^i, \ldots, a_T^i)$, with $a_t^i \in \mathcal{V}_A$, where $\mathcal{V}_A$ is the motion vocabulary.

4.2 Recap: the SMART framework

SMART is a decoder-only transformer that jointly models multi-agent trajectories and road structures as discrete sequences. It is trained using two next-token prediction tasks: one for road vectors and one for agent motions.

Road next-token prediction. Given a sequence of road tokens $r_{0:j}$, SMART predicts the next road token r_{j+1} according to:

$$p_\gamma(r_{j+1} \mid r_{0:j}), \tag{4.2}$$

with the loss function:

$$\mathcal{L}_{\text{road}}(\gamma) = -\sum_{j=0}^{J-1} \log p_\gamma(r_{j+1}^{\text{gt}} \mid r_{0:j}). \tag{4.3}$$

Motion next-token prediction. For each agent i, SMART predicts the next motion token a_{t+1}^i conditioned on its own past tokens, other agents, and road tokens:

$$p_\theta(a_{t+1}^i \mid a_{0:t}^i, \{a_{0:t}^j\}_{j \neq i}, R), \tag{4.4}$$

with the corresponding loss:

$$\mathcal{L}_{\text{motion}}(\theta) = -\sum_{i=1}^{N_A} \sum_{t=0}^{T-1} \log p_\theta(a_{t+1}^{i,\text{gt}} \mid a_{0:t}^i, \{a_{0:t}^j\}_{j \neq i}, R). \tag{4.5}$$

Training objective. The global objective of SMART is:

$$\mathcal{L}(\theta, \gamma) = \mathcal{L}_{\text{road}}(\gamma) + \mathcal{L}_{\text{motion}}(\theta). \tag{4.6}$$

At inference time, trajectories are generated autoregressively:

$$a_{t+1}^i \sim p_\theta(\cdot \mid a_{0:t}^i, \{a_{0:t}^j\}_{j \neq i}, R). \tag{4.7}$$

Attention mechanism. Each agent embedding e_t^i is updated via a stack of attention layers, applied in a fixed order: temporal self-attention across the agent's past tokens $(e_0^i, \ldots, e_t^i)$, followed by cross-attention to road tokens R, and finally self-attention across all agent embeddings $(e_t^1, \ldots, e_t^{N_A})$ at the current timestep. Formally, the update is:

$$e_t^i \leftarrow \mathcal{A}_{\text{agent}}\Big(\mathcal{A}_{\text{map}}\big(\mathcal{A}_{\text{temp}}(e_t^i)\big)\Big), \tag{4.8}$$

where each $\mathcal{A}$ denotes a multi-head attention module. These layers successively incorporate temporal dynamics, environmental context, and social interactions into the agent representation.

4.3 Difficulty representation via VAE

To condition SMART on the difficulty of trajectories, we map difficulty matrices D^i into a continuous latent space. We adopt a *Variational Autoencoder* (VAE) architecture [**?**].

Encoder. The encoder network maps a difficulty matrix D^i to the parameters of a Gaussian posterior:

$$q_\phi(z^i \mid D^i) = \mathcal{N}(\mu_\phi(D^i), \Sigma_\phi(D^i)), \quad z^i \in R^Z, \tag{4.9}$$

where μ_ϕ and Σ_ϕ are outputs of neural networks parameterized by ϕ.

Decoder. The decoder reconstructs the difficulty matrix from the latent code:

$$p_\psi(D^i \mid z^i). \tag{4.10}$$

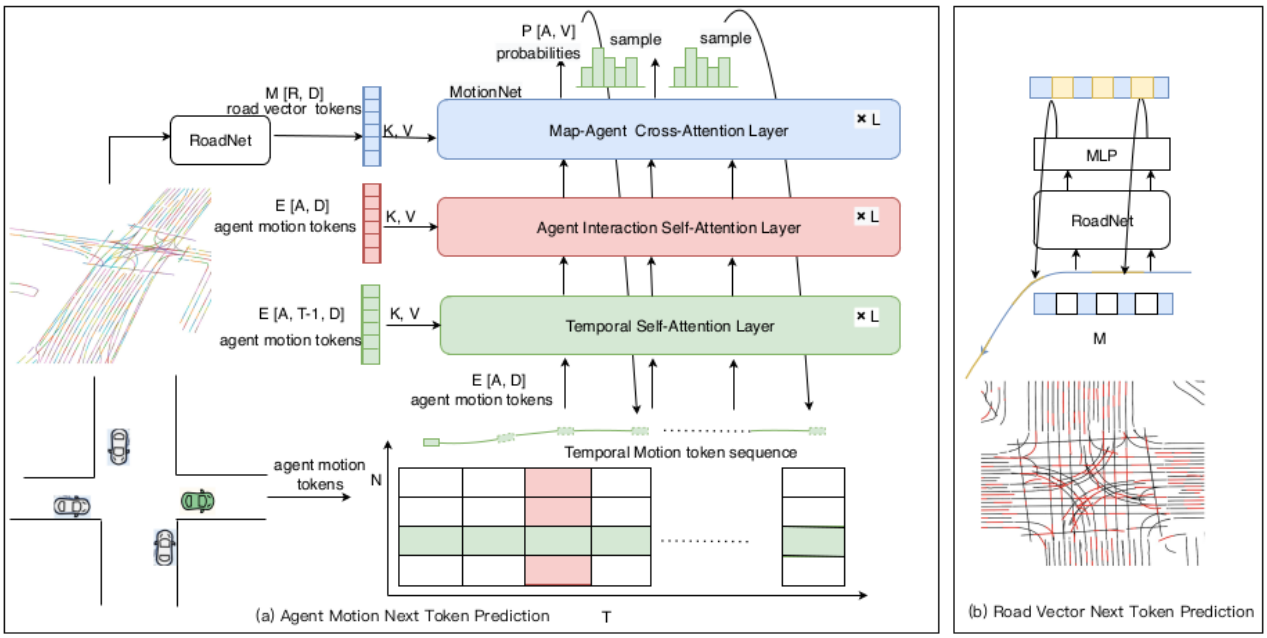


Figure 4.1: **(Extracted from SMART [58])** The architecture of SMART framework (a) We train a decoder-only transformer that predicts the motion tokens of multi-agents conditional on previous motion tokens, interactive agent motion tokens, and encoding road tokens. The model is trained to predict the next motion token. (b) Illustration for our proposed road spatial understanding training task.

Training objective. The VAE is trained to minimize:

$$\mathcal{L}_{\text{VAE}} = E_{q_\phi(z^i|D^i)}[-\log p_\psi(D^i \mid z^i)] + \beta\,\text{KL}(q_\phi(z^i \mid D^i) \,\|\, \mathcal{N}(0, I)), \tag{4.11}$$

where the first term enforces reconstruction of the difficulty metrics, and the second term regularizes the latent distribution toward an isotropic Gaussian prior. The hyperparameter β balances the two terms.

This training ensures that z^i is both informative about the difficulty of D^i and smoothly distributed in latent space, which is crucial for sampling during inference.

4.4 Difficulty-conditioned SMART

We now integrate the difficulty latents into SMART. In the original model, each agent token at time t is represented by an embedding $e_t^i \in R^D$. To condition on difficulty, we define:

$$h^i = g_\omega(z^i) \in R^D, \tag{4.12}$$

where g_ω is a multi-layer perceptron mapping latent codes to embedding space. The conditioned embedding is then:

$$\tilde{e}_t^i = e_t^i + h^i. \tag{4.13}$$

The motion prediction distribution becomes:

$$p_\theta(a_{t+1}^i \mid \tilde{e}_{0:t}^i, \{\tilde{e}_{0:t}^j\}_{j\neq i}, R). \tag{4.14}$$

This design allows SMART to incorporate difficulty as an auxiliary conditioning signal, effectively guiding the generative process to produce trajectories aligned with the desired difficulty level.

By injecting our modification prior to the context update (see 4.8), the latent vector influences the entire embedding refinement process — temporal history, social interactions, and road context — enabling the generation of trajectories aligned with the desired difficulty level.

4.5 Difficulty-controlled generation

At inference time, we can directly manipulate the latent variables:

1. Sample $z^i \sim \mathcal{N}(0, I)$ or from a chosen region of latent space corresponding to a desired difficulty profile.
2. Map to difficulty embeddings $h^i = g_\omega(z^i)$.
3. Run SMART autoregressively with conditioned embeddings $\tilde{e}_t^i$.

The resulting trajectories are:

$$\tau^i \sim p_\theta(\cdot \mid \mathcal{C}, z^i). \tag{4.15}$$

By varying z^i, we can smoothly control the difficulty of simulated driving scenarios, enabling the generation of synthetic data that spans from simple, low-risk interactions to challenging, high-difficulty maneuvers.

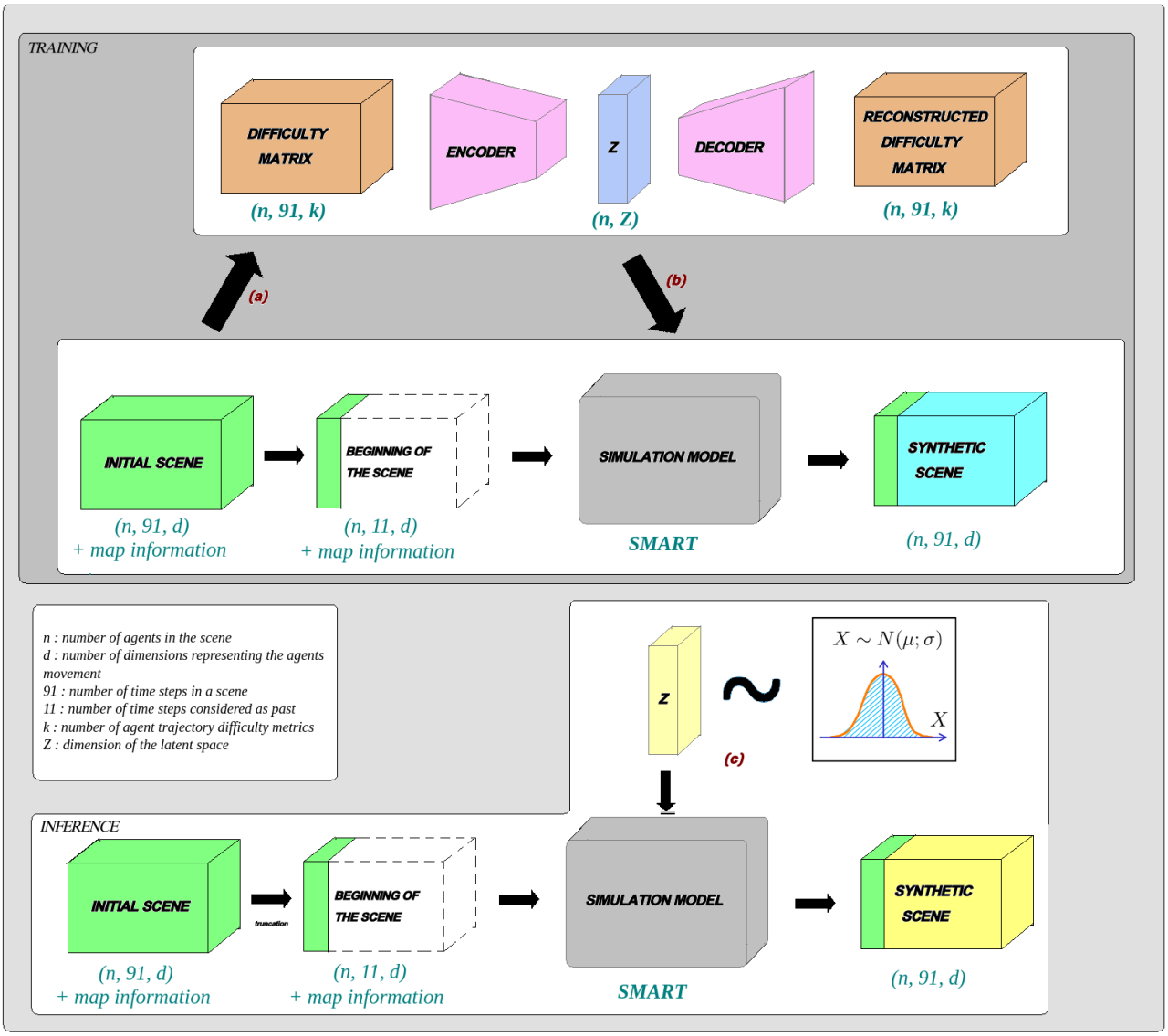


Figure 4.2: Training and Inference processes

Training

(a) From the initial scene information, we compute a *difficulty matrix* for each agent by evaluating predefined metrics over its trajectory. Each matrix is then passed through a pretrained VAE encoder to obtain a latent vector z^i that compactly represents the difficulty characteristics of agent i's trajectory.

(b) The set of latent vectors $\{z^i\}_{i=1}^{N_A}$ is then provided as an additional conditioning signal to the SMART model. Concretely, these vectors are mapped through an MLP and injected into the agent embeddings, allowing SMART to incorporate difficulty information during training and align generated behaviors with the observed difficulty patterns.

Inference

(c) At inference time, we can sample latent vectors z^i directly from the prior distribution in the latent space (e.g., a standard Gaussian). These samples act as controllable difficulty codes: by varying them, we condition SMART to generate synthetic scenarios with adjustable levels of difficulty, without requiring explicit recomputation of difficulty metrics from real trajectories.

Algorithm 1 Train SMART with Difficulty Conditioning (transfer-learning or not)

Inputs: Dataset $\mathcal{S}$ with tokens and precomputed $\{D^{(n),i}\}$; pretrained VAE encoder Enc_ϕ; conditioning MLP g_ω; SMART parameters Θ

Outputs: Trained parameters (Θ, ω)

Initialization:
  Option 1: load $\Theta \leftarrow \Theta_{\text{SMART-pretrained}}$
  Option 2: initialize Θ randomly from scratch
initialize g_ω randomly
Training:
for each training epoch **do**
  for each batch of scenarios $\{S^{(n)}\}$ **do**
    for each scenario $S^{(n)}$ **do**
      tokenize road network $\rightarrow R^{(n)}$
      tokenize agent trajectories $\rightarrow \{a_{0:T}^{(n),i}\}$
      for each agent i in $S^{(n)}$ **do**
        retrieve difficulty matrix $D^{(n),i}$
        compute latent vector $z^{(n),i} \leftarrow \text{Enc}_\phi(D^{(n),i})$
        map to embedding $h^{(n),i} \leftarrow g_\omega(z^{(n),i})$
        for $t = 0$ to T **do**
          base embedding $e_t^{(n),i} \leftarrow \text{Embed}(a_t^{(n),i})$
          conditioned embedding $\tilde{e}_t^{(n),i} \leftarrow e_t^{(n),i} + h^{(n),i}$
        end for
      end for
      run SMART decoder with conditioned embeddings and road tokens
      predict next-token distributions $\hat{p}_\Theta(a_{t+1}^{(n),i} \mid \cdot)$
      compute motion loss $\mathcal{L}_{\text{motion}}$ (cross-entropy with ground truth)
      compute road loss $\mathcal{L}_{\text{road}}$ as in original SMART
      total loss $\mathcal{L} = \mathcal{L}_{\text{motion}} + \mathcal{L}_{\text{road}}$
    end for
    Option A: update both Θ and ω by gradient descent
    Option B: freeze Θ and update only ω
  end for
end for

Chapter 5

Experiments and Results

This section presents the detailed implementations, results, and ongoing experiments.

5.1 Experimental setup

5.1.1 Dataset

The Waymo Open Motion Dataset will serve as the foundation for all our experiments in this report [54]. This dataset provides access to high-quality, real-world driving scenarios represented in a bird's-eye-view format, which is particularly well-suited for mid-to-end autonomous driving research paradigms. The dataset provides a rich set of features describing the road environment, traffic signals, and dynamic agents within each scenario. These features are summarized below, grouped by their main categories:

- **Roadgraph features:** Include sampled points from map elements (e.g., lane centers, road edges, crosswalks) with their 3D coordinates, direction vectors, semantic types, and validity flags. These features represent the static environment geometry.
- **Scenario features:** A unique scenario identifier is provided to distinguish different recorded driving scenes.
- **Agent state features:** For each object in the scene, the dataset provides its trajectory information across past, current, and future time steps. This includes positions, velocities, bounding box sizes, headings, and validity masks. Additional flags indicate the self-driving car (SDC), interactive agents, and objects selected for prediction.
- **Traffic light features:** The state of traffic lights is recorded over time, including their categorical state (e.g., stop, go, caution), positions in global coordinates, the associated lane IDs, and validity flags. These features are available for past, current, and future steps.

Together, these elements form a comprehensive representation of both the static road context and dynamic traffic participants, enabling learning-based methods to model motion prediction and interaction within realistic driving scenarios.

Each scenario has a duration of 9 seconds and is sampled at a frequency of 10 Hz, resulting in 91 time steps. The goal of our experiment is to use the first 11 time steps—corresponding to 1 second of past context plus the current state—to predict the subsequent 80 time steps (i.e., the remaining 8 seconds of the scenario). The predictions should be generated in a manner that is both realistic and subject to challenging constraints. This formulation follows the Waymo Open Sim Agents Challenge [56], with the addition of conditioning.

A complete description of the dataset is provided in the Appendix 7.5 for reference.

5.1.2 SMART Architecture and Latent-Conditioned Extension

The SMART [58] implementation that we build upon is taken from the framework of *Closed-Loop Supervised Fine-Tuning of Tokenized Traffic Models* [2]. Different versions of SMART exist according to their parameter count (1M, 7M, and 100M). In this work, we use the 7M parameter variant as a balance between computational efficiency and representational capacity.

Overview of SMART

SMART (**S**calable **M**ulti-agent **A**uto-**R**egressive **T**rajectories) is a Transformer-based architecture that formulates motion prediction as an autoregressive token-generation problem. Instead of directly regressing future continuous trajectories, SMART discretizes both the motion of traffic participants and the map elements into vocabularies of tokens. Future behavior is then generated step by step by predicting the next token, in the same spirit as language models.

The architecture is composed of two main components:

- **Map encoder.** Road geometry and semantics are encoded into vectorized tokens. A self-attention mechanism over these map tokens captures spatial structure and topological relations.
- **Motion decoder.** Agent trajectories are represented as motion tokens. For each agent and prediction step, the decoder applies temporal self-attention (capturing dynamics across time), cross-attention from map to agent (capturing road context), and agent–agent self-attention (capturing interactions). This process is repeated in a stack of layers, and the model predicts the next motion token for each agent.

By formulating prediction as next-token classification, SMART avoids distribution-fitting complexities and can naturally represent multiple modes of behavior. Training is performed with a cross-entropy loss over the discrete token space, and inference proceeds autoregressively by rolling out tokens into continuous trajectories.

Latent-Conditioned Extension

While the original SMART model bases its predictions only on map context and agent history, our extension introduces an additional conditioning signal in the form of a learned latent representation. The goal of this modification is to steer the model toward producing behaviors that are not only consistent with the environment but also consistent with higher-level, scenario-specific styles such as cautious, aggressive, or smooth driving.

Latent representation. The latent vector is obtained from a separate model trained to summarize agent-level kinematic properties (for example, accelerations). This vector captures hidden factors of motion style and compresses them into a compact representation.

Integration into SMART. The latent vector is projected into the same feature space as the agent embeddings and then combined with the token-level trajectory representations. This enriched representation is passed through the same sequence of temporal, map-to-agent, and agent–agent attention layers as in the original SMART architecture. In this way, every predicted step is influenced not only by past motion and map context but also by the global latent style.

Training strategy. To preserve the strengths of the pretrained SMART model, the majority of the network weights are kept frozen. Only the small adapter that maps the latent representation into the feature space, and the fusion layer that merges this signal with the agent embeddings, are updated during training. The objective remains identical to the original formulation: cross-entropy loss over the next motion token.

Motivation and Benefits

The proposed extension has several advantages:

- **Minimal architectural change.** Conditioning is introduced additively, without altering the attention structure or the overall design of SMART. This keeps inference efficient and scalable.
- **Behavioral control.** The latent captures consistent scenario-level characteristics, enabling controllable generation of agent behaviors beyond what is encoded in map and history alone.
- **Data efficiency.** Because only a small adapter module is trained, the approach is lightweight and can adapt to new data or tasks with limited additional supervision.
- **Consistency.** The latent is provided during both training and inference, ensuring that conditioned behavior generalizes to rollout scenarios.

Other model to use as a foundation

Other models has been thought of as interesting to condition. Among them, we can list :

- CLSFT : Among the current SoTA on the Waymo Open Sim Agents Challenge [65]
- Wayformer : 2022 best model on the Waymo Open Sim Agents Challenge [36]
- MTR : Transformer-based joint motion prediction model [21]
- Autobot : Autoregressive transformer for multi-agent trajectories [23]
- Diffusion models

5.1.3 Baselines implementation

Intelligent Driver Model

One of the baselines used to compare our approach is the Intelligent Driver Model (IDM) [48], which was briefly introduced in the **Related Works Section** 2.1. A more detailed overview of its implementation is provided in the **Appendix** 7.1.2.

It is important to note that IDM was not originally designed as a scenario generation algorithm and thus is not intended to control every agent in a scene. Nevertheless, as a fundamental rule-based method, it provides a useful baseline for comparison.

IDM is a speed regulation model that computes the future absolute velocity of a vehicle based on its current speed, the speed and distance of the leading vehicle, as well as intrinsic vehicle and road parameters. It was first proposed by Treiber, Hennecke, and Helbing in 2000. The update rule is defined as:

$$v_{t+1} = v_t + \Delta t \cdot u_t$$

with

$$u_t = \max\left(a\Big(1 - \Big(\frac{v_t}{v_0}\Big)^{\delta} - \Big(\frac{s^*}{s}\Big)^2\Big), -B\right),$$

$$s^* = s_0 + \max\left(0, v_t T + \frac{v_t \Delta v}{2\sqrt{ab}}\right).$$

This implementation differs from the original formulation in [48] by introducing a maximum braking acceleration B, which prevents divergence issues inherent in the original model. Since our experiments involve simulating tens of millions of agents, divergence inevitably occurred, and this patch proved necessary. While other works [68, 6] propose more advanced improvements to IDM, this adjustment was sufficient for our needs.

Another crucial aspect of rule-based methods is pathway selection. Some models, such as PDM [17], incorporate realistic decision-making mechanisms for direction control. For simplicity, our implementation retains IDM's original assumption of keeping vehicles near the road center, with interpolation ensuring smooth transitions toward the desired path and avoiding unrealistic maneuvers.

When reaching the end of a road, a connected road must be selected as the next segment. Two strategies were implemented:

1. **Random selection.** The next road is sampled uniformly at random from the set of available road segments connected to the current one.

2. **Best path selection.** To avoid penalizing IDM for unrealistic road choices, this method leverages ground-truth trajectories available in our validation dataset. By using the final positions of agents' trajectories, the shortest roadlines set closest to the ground-truth endpoint is selected. This ensures more logical paths and better realism profiles when evaluated with metrics such as the **Average Displacement Error (ADE)**. **ADE measures the average distance between predicted and ground-truth trajectories**, and is therefore highly sensitive to errors in road selection. Without this correction, IDM would be unfairly penalized for poor routing rather than for its actual driving style. While this strategy is technically unrealistic—since it exploits knowledge of the ground-truth scenario—it aligns with the fact that most realism metrics themselves require a reference trajectory. Hence, this formulation isolates IDM's evaluation to its *driving style*, rather than its inability to perform high-level route planning.

Others

Other models to compare the method to are discussed in the **Future Work and Prospects** chapter as they have not been implemented yet.

5.1.4 Difficulty metrics over agent trajectory

Over the course of our reflection, several *per-agent* metrics emerged as potential candidates for conditioning driving behaviors. These metrics are particularly appealing because they can be defined without requiring access to an original ground-truth scenario, making them usable in more general setups. Our analysis focused on two main questions: (i) would conditioning on these metrics result in meaningful and diverse behaviors, and (ii) is it feasible to condition on them given the characteristics of our dataset ? The following list summarizes the selected metrics and our considerations:

- **Lane center distance.** Conditioning on the distance between the vehicle's center and the nearest lane center could encourage behaviors such as drifting away from the lane center or frequent lane changes. These behaviors are particularly interesting because they directly increase scenario difficulty, while also reflecting realistic variations observed in naturalistic driving data. Since imperfect lane following and spontaneous lane changes are relatively common in the dataset, conditioning on this metric appears both feasible and valuable.

- **Comfort.** Comfort can be approximated through simple physical quantities such as angular speed, acceleration, or jerk. These low-level features strongly influence the driving style: for example, conditioning on highly fluctuating acceleration could lead to aggressive stop-and-go behaviors, producing uncomfortable and potentially dangerous motion patterns. Because variations in acceleration and angular speed are frequent in the dataset, conditioning on comfort-related measures is both practical and promising.

- **Off-road.** Conditioning on whether the vehicle should remain on the drivable surface or deviate off-road would provide a direct handle on generating unsafe behaviors. However, off-road events are extremely rare in the dataset, as they are uncommon in real driving and often excluded from curated logs. This rarity makes it difficult for data-driven models (both embeddings and generative models) to properly capture and reproduce such behaviors.

- **Collision.** Similarly, conditioning on collisions would allow explicit generation of adversarial scenarios where vehicles are forced into accidents, creating strong stress cases for downstream driving policies. Yet collisions are also extremely rare, or almost entirely absent, in large-scale public datasets. This scarcity stems not only from their rarity in real-world traffic, but also from dataset curation practices—data providers are unlikely to release collision-heavy scenarios for legal and ethical reasons. Consequently, while theoretically attractive, conditioning on collisions is impractical with the available data.

In summary, while off-road and collision conditioning remain largely impractical due to their scarcity in real-world datasets, **lane center distance and comfort** emerge as the most promising metrics, offering both feasibility and the ability to induce diverse, realistic, and challenging behaviors.

5.1.5 Scenario metrics : Realism and difficulty

Waymo Open Sim Agents Metrics

The Waymo Open Sim Agents Challenge [56] evaluates the quality of traffic simulation by comparing generated agent behaviors against real-world trajectories. Since simulation must reflect not only individual motion but also multi-agent consistency and interaction, the evaluation is based on a suite of behavior-characterizing metrics. These metrics are computed over rollouts of all agents in the scene and aggregated into a meta-metric used for leaderboard ranking.

The metrics fall into three main categories:

- **Agent motion.** These features measure the consistency of simulated motion with realistic kinematic profiles. They include linear speed, angular speed, linear acceleration, and angular acceleration. Unrealistic fluctuations in these quantities directly reduce the realism score.

- **Agent interactions.** These features evaluate how simulated agents interact with each other. They include collision rate, minimum distance to the closest object, and time-to-collision (for vehicles). Properly capturing interactions is crucial for reproducing realistic multi-agent traffic scenes.
- **Map adherence.** These features assess how well simulated agents comply with map constraints, such as staying on the drivable surface and obeying traffic signals. Metrics include off-road rate, distance to road edge, and traffic light violation rate (for vehicles).

Submissions are scored by sampling multiple stochastic rollouts for each scenario (typically 32) and estimating the likelihood of the logged ground-truth trajectory under the simulated distribution (see figure 5.1). For each feature category, histograms or kernel density estimations are used to approximate the distribution of simulated behaviors, and the likelihood of the real behavior is computed.

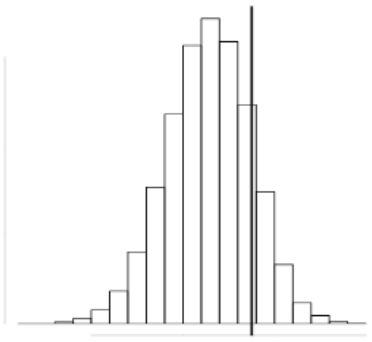


Figure 5.1: A schema of the likelihood computed over the histogram of the rollouts

Finally, all metrics are combined into a weighted **meta-metric**, which serves as the basis for leaderboard ranking. Higher values indicate simulations that better reproduce the diversity, realism, and feasibility of real-world traffic, while still capturing rare events. This framework has become one of the standards for evaluating generative multi-agent traffic models. More informations available in the Appendix 7.1.1.

Controllability and Realism Metrics

Recent works in scenario generation emphasize two central criteria: the **controllability** of generated scenarios, i.e., how effectively a model can produce safety-critical situations on demand, and their **realism**, i.e., how well they resemble naturalistic driving behaviors. These criteria are particularly highlighted in the evaluation of CCDiff [34].

Controllability. Controllability is measured through the **Scenario-wise Collision Rate (SCR)** [67, 45]. This metric calculates the proportion of test scenarios in which at least one collision occurs between different agents. To ensure comparability across models, SCR is standardized into a **Controllability Score (CS)** ranging from 0 to 1, where higher values indicate stronger ability to generate challenging, safety-critical interactions.

Realism. Quantifying realism remains more difficult, as it requires capturing nuanced statistical properties of driving trajectories. Following prior work [22, 45, 7], CCDiff evaluates realism with three complementary metrics:

- **Off-Road Rate (ORR)**: the proportion of trajectories leaving the drivable area, a direct indicator of physical plausibility.

- **Final Displacement Error (FDE)**: the distance between the final positions of generated and reference trajectories, reflecting predictive accuracy.
- **Comfort Distance (CFD)**: a measure of trajectory smoothness, penalizing unrealistic accelerations or jerky motions.

These measures are standardized and averaged into a single **Realism Score (RS)**, also normalized between 0 and 1, with higher values corresponding to more human-like behaviors.

Joint Evaluation. Because controllability and realism often trade off against each other, evaluations consider both simultaneously using multi-objective optimization tools such as **Generational Distance (GD)** and **Inverted Generational Distance (IGD)**. These metrics assess how close a method lies to the Pareto frontier balancing realism and controllability [14] (see figure 5.2).

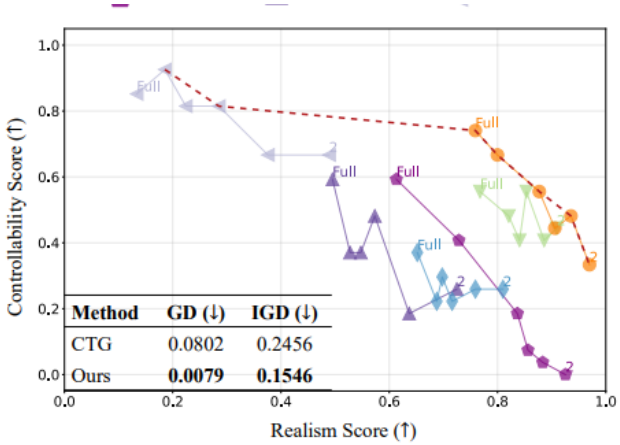


Figure 5.2: An example of Pareto frontier (extracted from CCDiff [34])

This dual framework provides a principled way to benchmark scenario generation models, ensuring that methods are not only capable of producing safety-critical cases on demand, but also remain faithful to realistic traffic dynamics. More informations available in the Appendix 7.1.2.

5.2 Results and on-going experiments

5.2.1 Embedding

The first step of our experiments was to design an effective embedding of the selected metrics in order to produce a latent vector capable of conditioning the generative model.

In this initial setup, we focused on two low-level metrics: **acceleration** and **angular speed**. As discussed in Section 5.1.4, these metrics strongly influence driving behavior while being straightforward to compute from the dataset.

Our embedding model was implemented as a **Variational Auto-Encoder (VAE)** [31].

A VAE is a probabilistic generative model that learns a latent representation z of input data x by combining an encoder and a decoder. Unlike classical auto-encoders, which map inputs deterministically to a latent vector, VAEs model the latent space as a distribution. Specifically, the encoder produces parameters of a Gaussian distribution, $\mu(x)$ and $\sigma(x)$, from which a latent code is sampled using the reparameterization trick:

$$z = \mu(x) + \sigma(x) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

The decoder then reconstructs the input from z, producing $\hat{x} = f_\theta(z)$. Training minimizes a loss that balances reconstruction accuracy with regularization of the latent space:

$$\mathcal{L}(x, \hat{x}) = \underbrace{\|x - \hat{x}\|^2}_{\text{Reconstruction loss}} + \beta \cdot \underbrace{D_{\text{KL}}\left(q_\phi(z|x) \,\|\, p(z)\right)}_{\text{KL divergence}}.$$

Here, $q_\phi(z|x)$ is the approximate posterior learned by the encoder, $p(z)$ is the prior distribution (typically $\mathcal{N}(0, I)$), and β is a weight controlling the regularization strength. This structure ensures that the latent space remains continuous, smooth, and suitable for conditioning downstream generative models.

Since our objective was to reduce a temporal matrix of size $T \times K$ into a latent vector of size Z, we initially implemented a transformer-based encoder to capture temporal dependencies. For the decoder, we adopted a simple multi-layer perceptron (MLP), and we also tested an alternative architecture using MLPs for both encoder and decoder.

Several observations emerged from these experiments:

- **Transformer vs. MLP.** Regardless of hyperparameter choices, transformer-based encoders consistently underperformed compared to their MLP counterparts (see fig 5.3). This outcome was somewhat surprising, as transformers appeared well suited to handling sequential temporal data, and as the transformers overfitted quickly. Nevertheless, for the sake of reliability in our embedding, we retained the MLP structure.

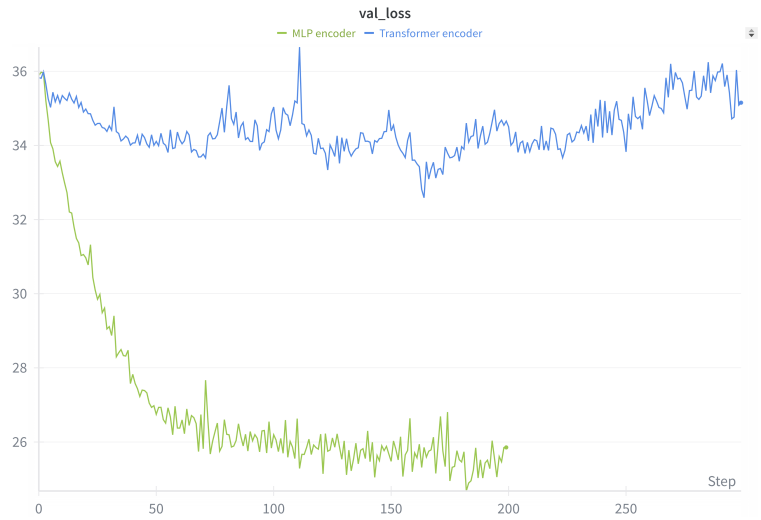


Figure 5.3: VAE loss comparison between MLP and Transformer encoders (best runs after hyperparameter optimization, on validation set).

- **Angular speed sparsity.** Angular speed values are predominantly zero, with only rare and small variations, even after normalization (see tab 5.1). This lack of variability makes the metric difficult to embed effectively, as rare data are poorly embedded by a VAE. Consequently, we simplified the embedding task to focus solely on acceleration, which is better distributed.

- **Acceleration anomalies.** Due to imperfections in the perception-to-bird's-eye-view pipeline, vehicle positions can occasionally "teleport" (see images 7.1), introducing spurious accelerations around $10g$ (see 5.1) (way higher than a rocket launch). To mitigate

Table 5.1: Statistical distribution of angular speed and acceleration. Angular speed shows a strong concentration of zero values, which motivated its removal from our embedding setup.

	Angular Speed	**Acceleration**
Mean	0.003772	0.049525
Std	0.216655	1.616612
Min	-15.689373	-9.996548
25%	-0.003435	-0.182926
50%	0.000000	0.000000
75%	0.004605	0.315053
Max	12.780204	9.994392
0.01 Quantile	-0.338012	-5.174864
0.10 Quantile	-0.025661	-1.446692
0.20 Quantile	-0.007716	-0.470842
0.30 Quantile	-0.000221	0.000000
0.40 Quantile	0.000000	0.000000
0.50 Quantile	0.000000	0.000000
0.60 Quantile	0.000000	0.000000
0.70 Quantile	0.000952	0.070820
0.80 Quantile	0.009391	0.648797
0.90 Quantile	0.032350	1.646728
0.99 Quantile	0.363532	5.314700

this issue, we clipped such extreme values by resetting them to zero. This follows the same strategy originally applied to masked (undetected) vehicles, ensuring consistency in preprocessing.

With these adjustments, the resulting embedding setup produces a latent space that is both well-regularized and compact in dimensionality (see fig 5.4), making it effective for conditioning purposes.

5.2.2 Conditionning

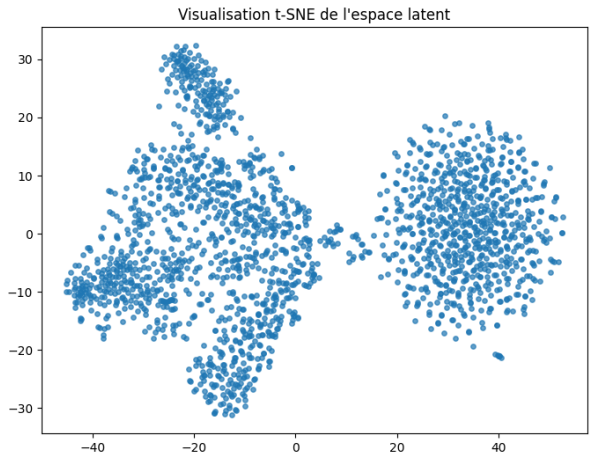
The latent vector used to conditioned our modified SMART (cond. SMART) is sampled in the latent space using a normal distribution.

Unfortunately, the preparation of this report interrupted the execution of this experiment, which has therefore not yet been launched.

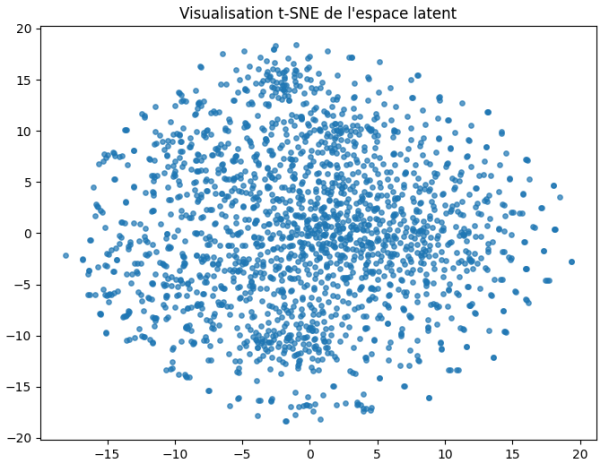
It is hoped that the initial steps of this experiment can be carried out before the submission deadline of this report, although this cannot be guaranteed.

The results table (see 5.2) will be filled in as the experiments are carried out.

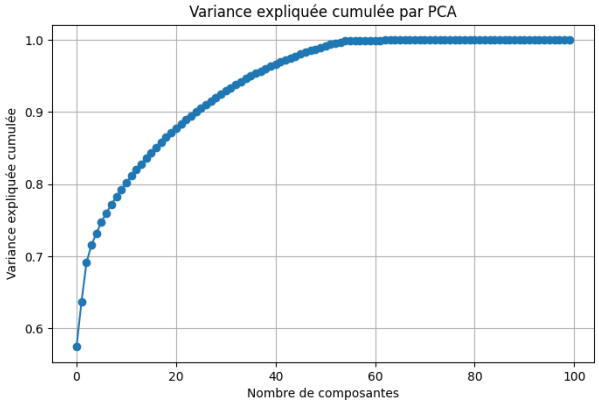
The latest results are not sufficient to draw firm conclusions, but they do indicate that the method leverages the difficulty–realism tradeoff, slightly increasing the collision rate while reducing displacement error performance.



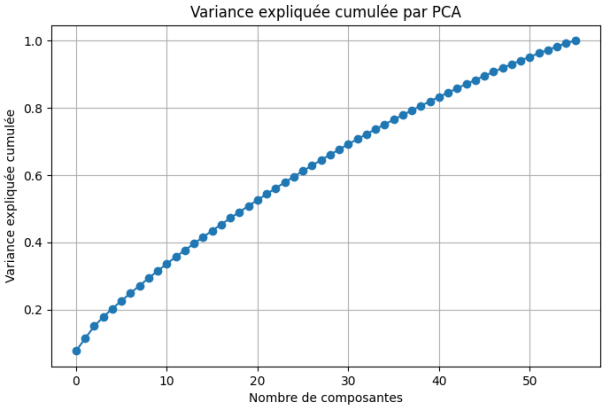
(a) t-SNE of latent space without VAE regularization



(b) t-SNE of latent space with VAE regularization



(c) Cumulative PCA eigenvalues on full latent space



(d) Cumulative PCA eigenvalues after removing useless dimensions

Figure 5.4: Latent space analysis of the acceleration embedding. (a–b) Visualization of the latent space using t-SNE, comparing the case without and with VAE regularization. (c–d) Cumulative explained variance ratios from PCA on the latent space, before and after removing unused dimensions to obtain a compact representation.

Table 5.2: Results on WOSAC and controllability/realism metrics. The likelihood metrics are not measured over the IDMs methods as they produce deterministic results on their driving, leading to an meaningless likelihood measurement. Lower (↓) is better for NLLs / ORR / FDE / CFD / GD / IGD; higher (↑) is better for CS / RS. "*" means extracted from the Waymo Open Sim Agents Challenge website.

Metric	**SMART**	**Cond. SMART**	**IDM (Random)**	**IDM (Best-Fit)**
Waymo Open Sim Agents — Component NLLs (per-metric likelihood; lower ↓ is better)				
Linear Speed Likelihood ↑	0.3646*	?	/	/
Linear Acceleration Likelihood ↑	0.4057*	?	/	/
Angular Speed Likelihood ↑	0.4231*	?	/	/

Continued on next page

Metric	**SMART**	**Cond. SMART**	**IDM (Random)**	**IDM (Best-Fit)**
Angular Acceleration Likelihood ↑	0.5845*	?	/	/
Distance to Nearest Object Likelihood ↑	0.3769*	?	/	/
Collision Likelihood ↑	0.9655*	?	/	/
Time-to-Collision (TTC) Likelihood ↑	0.8318*	?	/	/
Distance to Road Edge Likelihood ↑	0.6590*	?	/	/
Off-road (Road Departure) Likelihood ↑	0.9363*	?	/	/
min Average Distance Error (minADE) (Bonus) ↓	1.5447*	?	?	?
Waymo Realism meta-metric (weighted Likelihoods) ↑	0.7511*	?	/	/
Controllability (higher ↑ is better)				
Scenario-wise Collision Rate (SCR) ↑	0.4748	0.4862	0.4292	0.4286
Controllability Score (CS) $[0, 1]$ ↑	?	?	?	?
Realism (atomic metrics lower ↓; composite higher ↑)				
Off-road Rate (ORR) ↓	?	?	?	?
Final Displacement Error (FDE) [m] ↓	3.0465	3.2139	32.7559	11.9997
Comfort Distance (CFD) ↓	?	?	?	?
Realism Score (RS) $[0, 1]$ ↑	?	?	?	?
Multi-objective Summary (trade-off; lower ↓ is better)				
Generational Distance (GD) ↓	?	?	?	?
Inverted Generational Distance (IGD) ↓	?	?	?	?

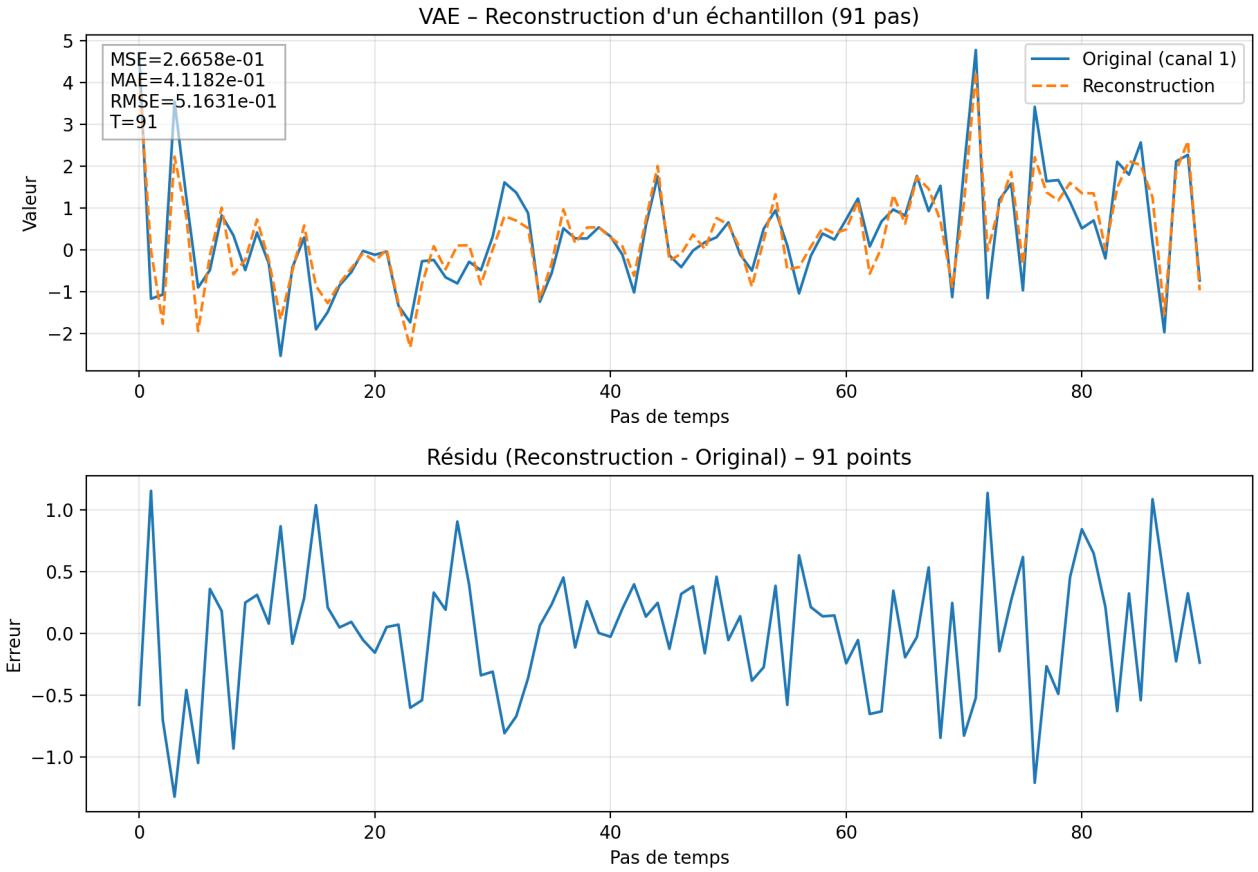


Figure 5.5: Visualization of the VAE performances
(a) Reconstruction of an acceleration sample thanks to the VAE
(b) Residual of the given reconstruction

Chapter 6

Future Work and Prospects

Due to the substantial workload, the long processing times associated with the scale of the dataset, the early stage at which this report was written during the internship, and the limited duration of the internship itself, a number of promising research directions could not be fully explored.

This section outlines avenues for further investigation, both as a continuation of the current methodology and as a critical reflection on how the present work could be extended and improved.

Training from Scratch Instead of Transfer Learning

In the current stage of experimentation, our approach relies on the *transfer learning* paradigm, where a pre-trained SMART model serves as the backbone that we subsequently condition to obtain our proposed model. This choice is primarily motivated by practical constraints: training SMART from scratch requires several days on multiple high-performance GPUs (e.g., 8 NVIDIA A100s), making it prohibitively costly within the scope of the internship.

Transfer learning is a reasonable strategy, as the additional conditioning signals are highly relevant to the task and should in principle guide the model toward generating scenarios that better capture the intended properties. However, this setup raises a potential risk: the model may fall into a local optimum where it effectively "ignores" the conditioning signals and reverts to replicating the original SMART behavior. In such a case, transfer learning could prevent the conditioning from being fully exploited, limiting the effectiveness of our approach.

A promising direction for future work would be to **train the model from scratch**, incorporating conditioning from the outset rather than as an adaptation of an already trained policy. Although computationally more expensive, this could ensure that conditioning is embedded as a fundamental part of the learned representation.

As intermediate alternatives, one could explore strategies such as:

- **Partial fine-tuning** — freezing only some layers of SMART while retraining others to better integrate conditioning.
- **Curriculum learning** — progressively introducing conditioning signals during training, making it harder for the model to ignore them.
- **Progressive conditioning** — gradually increasing the strength or complexity of conditioning objectives over the course of training.

Such approaches may provide a compromise between the efficiency of transfer learning and the robustness of training fully from scratch.

Investigating the Transformer Structure

As noted previously, the use of Transformer encoders within the VAE framework has not yet been thoroughly explored. Preliminary experiments suggested that, in the simplest setting, a Multi-Layer Perceptron (MLP) encoder outperformed the Transformer counterpart. However, this observation should not be interpreted as definitive. Transformers are well known for their ability to model sequential and long-range dependencies, which makes them particularly suitable for complex temporal embeddings such as those required in this work.

Future investigations should therefore examine Transformer-based encoders more deeply, especially in richer setups involving multiple metrics or more diverse conditioning signals. It is likely that the advantages of self-attention mechanisms will become more apparent in higher-dimensional or more heterogeneous data contexts. A systematic study comparing architectural choices (e.g., encoder depth, number of attention heads, positional encodings) and their impact on embedding quality would be a natural next step toward refining the experimental pipeline.

In particular, several concrete experiments could provide valuable insights:

- **Multi-metric embeddings.** Extending the VAE to jointly embed several per-agent metrics (e.g., acceleration, lane-center distance, comfort indicators) rather than focusing on a single variable could better exploit the Transformer's strength in handling parallel and correlated sequences.
- **Hierarchical architectures.** Introducing hierarchical VAEs or multi-scale Transformers could capture both short-term dynamics (e.g., acceleration) and long-term dependencies (e.g., lane switch) within a unified embedding space.
- **Generalization across datasets.** Evaluating Transformer embeddings on multiple benchmarks (e.g., nuScenes, Argoverse, Waymo) would test their robustness and help determine whether architectural complexity provides consistent benefits beyond a single dataset.

These directions highlight that, while initial experiments suggested limited value in Transformers for the simplest case, their potential remains significant for more demanding settings. Exploring these ideas would contribute to a more powerful and general embedding model, which could in turn strengthen the conditioning capabilities of future scenario generation pipelines.

Expanding the Set of Baselines

At present, the evaluation of our method relies primarily on comparisons with the original SMART model and with two variants of IDM. This choice was partly dictated by time constraints, as the implementation of conditioning-based scenario generation methods could not be completed within the internship timeframe. While these baselines are coherent—since neither SMART nor IDM is explicitly designed to generate conditioned or adversarial scenarios—they do not fully reflect the state of the art in this research direction.

Future work should therefore expand the set of baselines to include methods that explicitly aim to generate challenging or safety-critical scenarios. For example, approaches such as **FREA** or **CollisionGen** [12, 51] focus directly on the targeted generation of difficult interactions, making them particularly relevant points of comparison. Such baselines would provide a more rigorous assessment of the strengths and weaknesses of our method in the context of controllable scenario generation.

In addition, another research intern effort within Valeo is currently dedicated to reproducing **CtRL-Sim** [42] and its results. Incorporating CtRL-Sim as a baseline in future comparisons would be highly valuable, since it represents a recent method designed to generate reactive and

controllable driving agents. A side-by-side evaluation with CtRL-Sim could highlight whether our approach offers advantages in terms of scalability, controllability, or realism, and would position our work more clearly within the broader landscape of scenario generation methods.

Overall, extending the baseline set is an essential step to validate the contribution of our approach. By situating it alongside both classical (IDM, SMART) and modern conditioning-based baselines (FREA, CollisionGen, CtRL-Sim), future evaluations could offer a fairer and more comprehensive perspective on its performance.

Using RL Egos to Measure Scenario Difficulty

Another important perspective that has not yet been exploited in this work is the use of reinforcement learning (RL) policies as a proxy to measure scenario difficulty. While handcrafted metrics (e.g., collision rate, off-road rate, comfort indicators) provide useful signals, they often fail to capture the nuanced challenges of a scene. A scenario such as an intersection governed by priority-to-the-right may appear simple in terms of basic statistics, yet it can be extremely challenging for an autonomous driving policy.

A more informative approach is to introduce an **RL ego agent** into the generated scenarios and evaluate its performance. By observing how a trained policy succeeds or fails in completing tasks without infractions, one can obtain a policy-driven estimate of scenario difficulty. This method ties the notion of difficulty directly to the challenges faced by realistic decision-making systems, rather than relying solely on predefined thresholds.

This direction is made particularly feasible by the development of the **V-Max** framework within Valeo. V-Max provides a scalable mid-to-end simulation environment where a wide range of RL policies can be trained and benchmarked. Importantly, V-Max relies on the **ScenarioMax** format, which is the same representation used in the present work. This compatibility means that scenarios generated here can be directly evaluated using policies trained on V-Max, without the need for costly data conversion.

Leveraging RL ego agents from V-Max to assess the generated scenarios would provide a richer and more realistic measure of difficulty. It would also open the door to adaptive scenario generation pipelines, where scenario hardness is defined not only by abstract metrics but by the actual ability of trained agents to handle them. This would represent a significant step toward aligning scenario generation with the ultimate goal of stress-testing autonomous driving policies.

Studying and Mapping the Latent Space

Since our method is based on a VAE, the latent representation naturally allows for sampling random conditioning vectors, which in turn produce diverse scenarios. In principle, if one wishes to reproduce a specific scenario characteristic (for instance, trajectories with high acceleration variability), the encoder can be used to map this behavior into a latent vector that conditions the generator accordingly. This property makes the VAE a powerful tool for targeted scenario control.

However, conditioning on more abstract notions such as a global "difficulty score" is more challenging. Unlike explicit per-agent metrics, difficulty is not trivially associated with a single latent dimension, and there is no straightforward mapping between difficulty levels and points in latent space. To address this, future work could focus on **analyzing and structuring the latent space**.

Several directions appear promising:

- **Latent space visualization and clustering.** By projecting latent vectors (e.g., via t-SNE or PCA) and correlating them with scenario-level metrics, one could identify regions of latent space associated with specific types of difficulty or behavior. This would provide a first "map" of where challenging scenarios are located.

- **Heat maps of scenario difficulty.** Difficulty scores could be computed for generated scenarios and then used to build a heat map over the latent space, highlighting which areas correspond to simpler versus more difficult scenarios.
- **Difficulty-conditioned generation.** A generative model could be trained to take as input a desired difficulty score and output a latent vector that, when decoded, produces scenarios of the corresponding difficulty. This would be similar in spirit to approaches such as CtRL-Sim [42], but applied at the latent representation level rather than at the generation level.

Exploring these directions would help bridge the gap between low-level conditioning signals (e.g., acceleration, lane distance) and high-level scene properties such as difficulty. Ultimately, mapping and controlling the latent space could enable a more interpretable and user-driven generation process, where scenario properties are selected directly and reliably.

From Per-Agent Metrics to Global Scenario Metrics

The current approach to conditioning relies primarily on **per-agent metrics**, such as acceleration, angular speed, lane-center distance, or off-road indicators. While these are easy to compute and directly influence the behavior of individual agents, they may not fully capture the complexity of a driving scene. Difficulties in autonomous driving often emerge not from isolated behaviors, but from the **interaction between multiple agents** and their collective relation to the environment. Even if the current structure of SMART implies agent cross-attention, global conditioning could be more relevant.

A promising direction for future work would therefore be to investigate **global scenario-level metrics**. These could describe the overall characteristics of the scene, such as:

- **Interaction complexity** — e.g., the number of yielding or crossing events, or the degree of conflict between agents' intended trajectories.
- **Global risk indicators** — aggregated measures such as the proportion of near-collisions or critical time-to-collision values across the whole scene.

Using such scenario-level metrics would represent a shift in paradigm: instead of conditioning on the behavior of a single agent, the model would be guided by high-level properties of the entire traffic scene. This global perspective could lead to more natural and diverse scenario generation, while also making it easier to align scenario difficulty with real-world challenges encountered by autonomous vehicles.

In the long term, combining per-agent and global metrics could provide a multi-scale conditioning strategy, where low-level behaviors are embedded consistently within high-level scene objectives, resulting in richer and more controllable scenario generation.

Exploring Alternative Conditioning Mechanisms

In the current approach, conditioning is applied in the simplest possible manner: by **adding the conditioning vector** to the tokenized representation of the agent. This design choice allowed for a straightforward implementation and a clear first evaluation of conditioning. However, addition alone may not be the most expressive way to integrate conditioning signals into the generative process.

Several alternative mechanisms could be explored in future work:

- **Concatenation.** Instead of adding conditioning features, they could be concatenated to the agent token embedding, allowing the model to preserve the original representation while explicitly attending to conditioning information.

- **Cross-attention.** Conditioning vectors could be treated as separate tokens and introduced into the Transformer architecture through cross-attention, enabling the model to learn richer interactions between scene features and conditioning signals.
- **Gating mechanisms.** Inspired by conditional language models, conditioning could be integrated through multiplicative gates that modulate the contribution of certain latent dimensions based on the conditioning input.
- **Adapter layers.** Lightweight conditioning modules inserted into the backbone (e.g., adapters or FiLM layers) could guide the generative process while minimizing the risk of overriding useful pre-trained representations.

Investigating these approaches would help determine whether more structured or disentangled conditioning mechanisms can lead to stronger control over generated scenarios, as well as better balance between realism and adherence to the conditioning objectives.

Incorporating Traffic Light State Inference

Another promising direction for improvement lies in the explicit modeling of traffic signal states. As highlighted in [61], the **Waymo Open Motion Dataset (WOMD)** suffers from severe issues regarding missing or inaccurate traffic light annotations: more than 70% of signal states are absent or unreliable. Recent work has introduced automated imputation and correction methods that leverage vehicle trajectories and transportation-domain knowledge to reconstruct realistic traffic signal states. By aligning agents' stopping and acceleration patterns with inferred light states, these approaches reduce the estimated red-light violation rate from 15.7% in the raw dataset to just 2.9%, thus substantially improving realism.

Integrating such traffic light inference into our framework would allow us to condition scenarios not only on vehicle-level metrics (e.g., acceleration, comfort, or off-road tendencies), but also on intersection-level dynamics. This opens the possibility to generate scenarios where the difficulty arises directly from traffic signal interactions, such as contested yellow phases or ambiguous left-turn permissions, which are both common sources of complexity in urban driving.

Chapter 7

Conclusion

This internship set out to explore controllable scenario generation for autonomous driving, with a particular focus on varying the difficulty of multi-agent interactions. The work began with a comprehensive review of datasets, simulators, and generative models, which helped identify both the potential and the current gaps in the field.

The main contribution was the design of a framework that embeds measurable agent-level metrics into a latent representation of difficulty, and integrates this representation into a state-of-the-art generative model. Through this mechanism, scenarios can be systematically shifted along a spectrum of difficulty, offering a practical way to enrich training and evaluation data with challenging but realistic cases.

Baselines such as SMART, CATK, and IDM were re-implemented to ground the experiments, and the embedding step was successfully completed, yielding a compact and regularized latent space. Although the conditioning experiments could not be finalized within the internship timeframe, the technical foundations are in place and the early results validate the feasibility of the approach.

The limitations of this work are mainly linked to its stage of completion: quantitative evaluation of conditioned scenarios and comparison with additional baselines remain to be carried out. Nonetheless, the methodology and tools developed during the internship constitute a solid starting point for continued research.

Looking ahead, the prospects outlined in this report open several promising directions, from training new architectures to integrating reinforcement learning agents for closed-loop testing. By advancing the controllability and realism of generated scenarios, these efforts contribute to the broader goal of building more robust and reliable decision-making systems for autonomous vehicles.

Bibliography

[1] Awesome-imitation-learning: A list of imitation learning ressources. `https://github.com/kristery/Awesome-Imitation-Learning`.

[2] Closed-loop supervised fine-tuning of tokenized traffic models github repository. `https://github.com/NVlabs/catk`.

[3] The ieee/cvf conference on computer vision and pattern recognition, 2025.

[4] International conference on robotics and automation, 2025.

[5] Valeo: car equipment manufacturer. `https://www.valeo.com/en/`, 2025.

[6] Saleh Albeaik, Alexandre Bayen, Maria Teresa Chiri, Xiaoqian Gong, Amaury Hayat, Nicolas Kardous, Alexander Keimer, Sean T McQuade, Benedetto Piccoli, and Yiling You. Limitations and improvements of the intelligent driver model (idm). *SIAM Journal on Applied Dynamical Systems*, 21(3):1862–1892, 2022.

[7] Apratim Bhattacharyya, Kelvin Xu, Daniel J Phillips, John Lambert, Nemanja Djuric, and Balaji Li, Zsolt Kira. BITS: Bi-level imitation for traffic simulation. In *ICCV*, 2021.

[8] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[9] Holger Caesar, Juraj Kabzan, Kok Seang Tan, Whye Kit Fong, Eric Wolff, Alex Lang, Luke Fletcher, Oscar Beijbom, and Sammy Omari. nuplan: A closed-loop ml-based planning benchmark for autonomous vehicles. *arXiv preprint arXiv:2106.11810*, 2021.

[10] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8748–8757, 2019.

[11] Valentin Charraut, Waël Doulazmi, Thomas Tournaire, and Thibault Buhet. V-max: A reinforcement learning framework for autonomous driving.

[12] Keyu Chen, Yuheng Lei, Hao Cheng, Haoran Wu, Wenchao Sun, and Sifa Zheng. Frea: Feasibility-guided generation of safety-critical scenarios with reasonable adversariality. *arXiv preprint arXiv:2406.02983*, 2024.

[13] Jie Cheng, Yingbing Chen, and Qifeng Chen. Pluto: Pushing the limit of imitation learning-based planning for autonomous driving. *arXiv preprint arXiv:2404.14327*, 2024.

[14] Carlos A Coello Coello. *Evolutionary algorithms for solving multi-objective problems*. Springer, 2007.

[15] Gustavo Claudio Karl Couto and Eric Aislan Antonelo. Generative adversarial imitation learning for end-to-end autonomous driving on urban environments. In *2021 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–7. IEEE, 2021.

[16] Marco Cusumano-Towner, David Hafner, Alex Hertzberg, Brody Huval, Aleksei Petrenko, Eugene Vinitsky, Erik Wijmans, Taylor Killian, Stuart Bowers, Ozan Sener, et al. Robust autonomy emerges from self-play. *arXiv preprint arXiv:2502.03349*, 2025.

[17] Daniel Dauner, Marcel Hallgarten, Andreas Geiger, and Kashyap Chitta. Parting with misconceptions about learning-based vehicle motion planning. In *Conference on Robot Learning*, pages 1268–1281. PMLR, 2023.

[18] Weijia Ding, Hongzhuo Liang, Ruihan Yang, Masayoshi Tomizuka, and Wei Zhan. Realgen: Retrieval augmented generation for controllable traffic scenarios. *arXiv preprint arXiv:2312.13303*, 2023.

[19] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017.

[20] Kockelman K Fagnant, D.J. Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations. 2015.

[21] Huikun Fan, Jianyuan Gao, Zhenhua Wu, Wei Ma, Yuxuan Wang, Mingyu Ding, Xiaosong Li, Yang Yang, and Wei Zhang. Mtr: Multi-agent motion transformer for joint motion prediction. In *European Conference on Computer Vision (ECCV)*, 2022.

[22] Alex Fang, Shentao Yang, Krishnamurthy Dvijotham, et al. Controllable traffic generation with diffusion models. In *NeurIPS*, 2023.

[23] Sriram Ghosh, Shubham Choudhury, Wilko Schwarting, Daniela Rus, and Javier Alonso-Mora. Autobot: Adaptive behavior generation with transformers for multi-agent trajectory prediction. In *International Conference on Learning Representations (ICLR)*, 2022.

[24] Cole Gulino, Justin Fu, Wenjie Luo, George Tucker, Eli Bronstein, Yiren Lu, Jean Harb, Xinlei Pan, Yan Wang, Xiangyu Chen, et al. Waymax: An accelerated, data-driven simulator for large-scale autonomous driving research. *Advances in Neural Information Processing Systems*, 36:7730–7742, 2023.

[25] Niklas Hanselmann, Katrin Renz, Kashyap Chitta, Apratim Bhattacharyya, and Andreas Geiger. King: Generating safety-critical driving scenarios for robust imitation via kinematics gradients. In *European Conference on Computer Vision (ECCV)*, 2022.

[26] Hendrickson C. Mangones S. Samaras C Harper, C. Potential increases in travel with autonomous vehicles for the non-driving, elderly and people with travel-restrictive medical conditions. *Transportation Research Part C: Emerging Technologies*, 72:1–9, 2016.

[27] Zhiyu Huang, Zixu Zhang, Ameya Vaidya, Yuxiao Chen, Chen Lv, and Jaime Fernández Fisac. Versatile behavior diffusion for generalized traffic agent simulation. *arXiv preprint arXiv:2404.02524*, 2024.

[28] Bernhard Jaeger, Daniel Dauner, Jens Beißwenger, Simon Gerstenecker, Kashyap Chitta, and Andreas Geiger. Carl: Learning scalable planning policies with simple rewards. *arXiv preprint arXiv:2504.17838*, 2025.

[29] Chiyu Max Jiang, Yijing Bai, André Cornman, Christopher Davis, Xiukun Huang, Hong Jeon, Sakshum Kulshrestha, John Lambert, Shuangyu Li, Xuanyu Zhou, et al. Scenediffuser: Efficient and controllable driving simulation initialization and rollout. *arXiv preprint arXiv:2412.12129*, 2024.

[30] Saman Kazemkhani, Aarav Pandya, Daphne Cornelisse, Brennan Shacklett, and Eugene Vinitsky. Gpudrive: Data-driven, multi-agent driving simulation at 1 million fps. *arXiv preprint arXiv:2408.01584*, 2024.

[31] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[32] Quanyi Li, Zhenghao Mark Peng, Lan Feng, Zhizheng Liu, Chenda Duan, Wenjie Mo, and Bolei Zhou. Scenarionet: Open-source platform for large-scale traffic scenario simulation and modeling. *Advances in neural information processing systems*, 36:3894–3920, 2023.

[33] Yueyuan Li, Mingyang Jiang, Songan Zhang, Wei Yuan, Chunxiang Wang, and Ming Yang. End-to-end driving in high-interaction traffic scenarios with reinforcement learning. *arXiv preprint*, 2024. arXiv:2410.02253.

[34] Haohong Lin, Xin Huang, Tung Phan, David Hayden, Huan Zhang, Ding Zhao, Siddhartha Srinivasa, Eric Wolff, and Hongge Chen. Causal composition diffusion model for closed-loop traffic generation. In *CVPR*, 2025.

[35] Yiren Lu, Justin Fu, George Tucker, Xinlei Pan, Eli Bronstein, Rebecca Roelofs, Benjamin Sapp, Brandyn White, Aleksandra Faust, Shimon Whiteson, et al. Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7553–7560. IEEE, 2023.

[36] Nigamaa Nayakanti, Rami Al-Rfou, Aurick Zhou, Kratarth Goel, Khaled S Refaat, and Benjamin Sapp. Wayformer: Motion forecasting via simple & efficient attention networks. *arXiv preprint arXiv:2207.05844*, 2022.

[37] NeurIPS Workshop Organizers. Machine learning for autonomous driving workshop. In *Proceedings of NeurIPS Workshops*, 2019.

[38] Varaiya. P. Smart cars on smart roads: problems of control. *IEEE Transactions on Automatic Control*, 38:195–207, 1993.

[39] Jonah Philion, Xue Bin Peng, and Sanja Fidler. Trajeglish: Learning the language of driving scenarios. *arXiv preprint arXiv:2312.04535*, 2023.

[40] Guanren Qiao, Guorui Quan, Jiawei Yu, Shujun Jia, and Guiliang Liu. Trafficgamer: Reliable and flexible traffic simulation for safety-critical scenarios with game-theoretic oracles. *arXiv preprint arXiv:2408.15538*, 2024.

[41] Luke Rowe, Roger Girgis, Anthony Gosselin, Liam Paull, Christopher Pal, and Felix Heide. Scenario dreamer: Vectorized latent diffusion for generating driving simulation environments. *arXiv preprint arXiv:2503.22496*, 2025.

[42] Luke Rowe, Ömer Güney Gürbüz, Felix Heide, Liam Paull, Christopher Pal, and Roger Girgis. Ctrl-sim: Reactive and controllable driving agents with offline reinforcement learning. *arXiv preprint arXiv:2403.19918*, 2024.

[43] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vladimir Guo, Yuning Zhou, Yi Chai, Benjamin Caine, Vineet Vasudevan, Wei Han, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2446–2454, 2020.

[44] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[45] Shashank Suresh, Yichuan Charlie Li, Wei Zhan, and Dorsa Sadigh. Trafficsim: Learning to simulate realistic multi-agent behaviors. In *CVPR*, 2021.

[46] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[47] Tesla. Autopilot and full self-driving capability. `https://www.tesla.com/autopilot`, 2023.

[48] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. Congested traffic states in empirical observations and microscopic simulations. *Physical review E*, 62(2):1805, 2000.

[49] Valeo. Valeo develops and industrializes automotive-grade lidar and autonomous driving solutions. `https://www.valeo.com/en//`, 2025.

[50] Valentin Charraut Valeo. Scenariomax. `https://github.com/valeoai/ScenarioMax`, 2025. Accessed: 2025-08-27.

[51] Zi Wang, Shiyi Lan, Xinglong Sun, Nadine Chang, Zhenxin Li, Zhiding Yu, and Jose M. Alvarez. Enhancing autonomous driving safety with collision scenario integration (collisiongen). *arXiv preprint arXiv:2503.03957*, 2025.

[52] Waymo. Waymo safety report: On the road to fully self-driving. `https://waymo.com/safety`, 2017.

[53] Waymo. Waymo open interaction prediction challenge. `https://waymo.com/open/challenges/2025/interaction-prediction/`, 2025. Accessed: 2025-08-27.

[54] Waymo. Waymo open motion dataset. `https://waymo.com/open/data/motion/`, 2025. Accessed: 2025-08-27.

[55] Waymo. Waymo open scenario generation challenge. `https://waymo.com/open/challenges/2025/scenario-generation/`, 2025. Accessed: 2025-08-27.

[56] Waymo. Waymo open sim agents challenge. `https://waymo.com/open/challenges/2025/sim-agents/`, 2025. Accessed: 2025-08-27.

[57] World Health Organization (WHO). Global status report on road safety 2018. 2018.

[58] Wei Wu, Xiaoxin Feng, Ziyan Gao, and Yuheng Kan. Smart: Scalable multi-agent real-time motion generation via next-token prediction. *Advances in Neural Information Processing Systems*, 37:114048–114071, 2024.

[59] Wei Wu, Shuyang Sun, Danfei Xu, and Jiajun Wu. Smart: Scalable multi-agent real-time motion generation via next-token prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[60] Xiangguo Xie, Yuchen Li, Bo Li, et al. Advdiffuser: Generating adversarial safety-critical driving scenarios via guided diffusion. *arXiv preprint arXiv:2410.08453*, 2024.

[61] Xintao Yan, Erdao Liang, Jiawei Wang, Haojie Zhu, and Henry X. Liu. Improving traffic signal data quality for the waymo open motion dataset. *arXiv preprint arXiv:2506.07150*, 2025.

[62] Yuan Yin, Pegah Khayatan, Éloi Zablocki, Alexandre Boulch, and Matthieu Cord. Regents: Real-world safety-critical driving scenario generation made stable. *arXiv preprint arXiv:2409.07830*, 2024.

[63] Chris Zhang, Sourav Biswas, Kelvin Wong, Kion Fallah, Lunjun Zhang, Dian Chen, Sergio Casas, and Raquel Urtasun. Learning to drive via asymmetric self-play. In *European Conference on Computer Vision (ECCV)*, 2024.

[64] Jiawei Zhang, Chejian Xu, and Bo Li. Chatscene: Knowledge-enabled safety-critical scenario generation for autonomous vehicles. *arXiv preprint arXiv:2405.14062*, 2024.

[65] Zhejun Zhang, Peter Karkus, Maximilian Igl, Wenhao Ding, Yuxiao Chen, Boris Ivanovic, and Marco Pavone. Closed-loop supervised fine-tuning of tokenized traffic models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5422–5432, 2025.

[66] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15222–15232, 2021.

[67] Qingwen Zhao et al. Lctgen: Language-based controllable traffic generation for autonomous driving. In *NeurIPS*, 2023.

[68] Shirui Zhou, Shiteng Zheng, Junfang Tian, Rui Jiang, et al. Twenty-five years of the intelligent driver model: Foundations, extensions, applications, and future directions. *arXiv preprint arXiv:2506.05909*, 2025.

Appendix

Appendix A - Scenario generation details

IDM implementation

7.1 Scenario Evaluation Metric Frameworks in Autonomous Driving

This appendix provides a detailed technical breakdown of two key evaluation frameworks for autonomous driving scenarios: (1) the **Waymo Open Sim Agents Challenge (WOSAC) metrics**, and (2) the **CCDiff controllability and realism metrics**. We explain the categories of metrics, their mathematical formulation (including distribution-matching and likelihood computations), and how they are aggregated or used in multi-objective evaluation. Tables and equations are included to summarize these metrics and their roles.

7.1.1 Waymo Open Sim Agents Challenge Metrics

The Waymo Open Sim Agents Challenge introduced a comprehensive evaluation methodology to quantify how *realistic* a traffic simulation is. The core idea is to treat realistic simulation as a **distribution matching** problem: a simulator should produce scenarios whose distribution matches that of real-world driving logs. Since the true distribution of human driving behavior is unknown analytically, WOSAC evaluates a simulator by measuring how likely the *real* logged scenarios would be under the simulator's generated distribution. In practice, this is done by computing an approximate **likelihood** of the real outcomes under the simulator's scenario distribution.

Component Metric Categories. Instead of evaluating likelihood in the extremely high-dimensional raw trajectory space (which would be intractable), the challenge breaks down each scenario's outcome into a set of interpretable *component metrics*. These fall into three categories:

- **Agent Motion (Kinematics):** individual agent dynamics, such as speed and acceleration.
- **Agent Interactions:** interactions between agents, including distances and collisions.
- **Map Adherence:** adherence to road geometry, such as staying on drivable surfaces.

Table 7.1 summarizes the nine component metrics. Each captures a different aspect of realism in the simulation. Evaluating them separately provides interpretability and avoids the curse of dimensionality.

Table 7.1: WOSAC component metrics, grouped by category (motion, interaction, map adherence). Each metric is evaluated per agent over time in simulation and compared to real data distributions.

Category	**Metric Name**	**Description**
Agent Motion	Linear Speed	Instantaneous linear speed of an agent (m/s).
	Linear Acceleration	Linear acceleration (m/s^2).
	Angular Speed	Yaw (angular) speed (rad/s).
	Angular Accel. Magnitude	Magnitude of angular acceleration (rad/s^2).
Agent Interactions	Distance to Nearest Object	Distance to the closest other agent (m).
	Collision Occurrence	Indicator of a collision (binary event).
	Time-to-Collision (TTC)	Predicted time until collision (s).
Map Adherence	Distance to Road Edge	Lateral distance to the road or lane edge (m).
	Road Departure	Indicator of leaving the drivable area.

Distribution Matching and likelihood Computation. Let N be the number of scenarios in the evaluation set. For each scenario i, denote by $o^{(i)}_{<t_0}$ the initial context (past at handover) and $o^{(i)}_{\geq t_0}$ the future trajectories. An ideal simulator assigns high probability to the real future $o^{(i)}_{\geq t_0}$ given $o^{(i)}_{<t_0}$. The negative log-likelihood (NLL) is:

$$\mathcal{L}_{\text{NLL}} \;=\; -\frac{1}{N}\sum_{i=1}^{N} \log\, q_{\text{sim}}\Big(o^{(i)}_{\geq t_0} \,\Big|\, o^{(i)}_{<t_0}\Big)\,. \tag{7.1}$$

Direct computation is infeasible, so outcomes are factorized into the component metrics. For each metric k and scenario i, the likelihood of the real metric value is approximated from S stochastic rollouts of the simulator. A histogram or kernel density estimate forms the empirical distribution. The real metric value's log-likelihood is then averaged across agents and timesteps.

Composite Metric and Aggregation. NLL values from all component metrics are combined into a composite score:

$$\mathcal{L}_{\text{composite}} \;=\; \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K} w_k\,\big(-\ln L_{n,k}\big)\,, \tag{7.2}$$

where $K = 9$, $L_{n,k}$ is the likelihood of scenario n's ground-truth metric value under the simulator for metric k, and w_k are weights (summing to 1). Higher weights are given to safety-critical metrics such as collisions and off-road events. This composite NLL is the final meta-metric used to rank simulators.

7.1.2 CCDiff Controllability and Realism Metrics

The **Causal Composition Diffusion Model (CCDiff)** framework evaluates scenario generation on two axes: *controllability* (ability to enforce desired events such as collisions) and *realism* (naturalness of behaviors). It defines the Controllability Score (CS) and Realism Score (RS), and further uses Pareto-based multi-objective metrics.

Controllability Score (CS). Controllability is measured by the **Scenario-wise Collision Rate (SCR)**: the fraction of scenarios where at least one collision occurs. This is normalized across methods into $[0, 1]$:

$$\text{CS} \;=\; \frac{\text{SCR} - \min(\text{SCR})}{\max(\text{SCR}) - \min(\text{SCR})}\,. \tag{7.3}$$

A higher CS indicates stronger controllability.

Realism Score (RS). Realism combines three metrics:

- **Off-Road Rate (ORR):** fraction of scenarios where an agent leaves the drivable area.
- **Final Displacement Error (FDE):** average distance between generated and real final positions.
- **Comfort Distance (CFD):** we use **Comfort Distance (CFD)** as a realism metric to quantify the smoothness of generated trajectories. CFD measures how close the distribution of *comfort-related features* (e.g., jerk or acceleration smoothness) in generated rollouts is to that of the dataset.

  Formally, let a_t denote the acceleration at time t with step size Δt. The *jerk magnitude* is defined as

  $$j_t \;=\; \frac{\|a_t - a_{t-1}\|}{\Delta t}.$$

  We then construct histograms of jerk values for generated trajectories and for the dataset, normalized as probability distributions $\hat{p}_{\text{gen}}(j)$ and $\hat{p}_{\text{data}}(j)$. CFD is defined as the *Wasserstein-1 distance* between them:

  $$\text{CFD} \;=\; W_1\big(\hat{p}_{\text{gen}}(j),\, \hat{p}_{\text{data}}(j)\big).$$

  A smaller CFD indicates smoother, more human-like motion profiles, and thus higher realism.

Each metric is normalized and inverted so lower values yield higher scores. The RS is:

$$\text{RS} \;=\; 1 \;-\; \frac{1}{3}\Big(\tilde{\text{ORR}} \;+\; \tilde{\text{FDE}} \;+\; \tilde{\text{CFD}}\Big)\,, \tag{7.4}$$

with $\tilde{m}$ denoting normalized values in $[0, 1]$. Higher RS indicates more realistic trajectories.

Pareto Front Analysis. Because realism and controllability can conflict, CCDiff employs multi-objective evaluation via:

$$\text{GD} = \left(\frac{1}{|D|}\sum_{d\in D}\min_{p\in P}\| d - p \|^q\right)^{1/q}, \tag{7.5}$$

$$\text{IGD} = \left(\frac{1}{|P|}\sum_{p\in P}\min_{d\in D}\| p - d \|^q\right)^{1/q}, \tag{7.6}$$

where D is the set of method outcomes in (RS, CS) space and P the Pareto-optimal set. GD (Generational Distance) measures closeness to the Pareto front; IGD (Inverted GD) measures coverage. Lower values are better for both.

Summary. CS evaluates the ability to generate desired events, RS measures naturalness of behaviors, and GD/IGD quantify trade-offs. Together, these metrics provide a rigorous framework for assessing both realism and controllability in generated scenarios.

Appendix B - Scenario generation details

IDM implementation

As a rule-based method, Intelligent-Driver Model (IDM) needs a precise description of the rules used in order to get a real understanding of the method.

Speed Calculus

IDM is a speed calculation method intending to infer the future absolute speed of the vehicle using the previous vehicle's speed, relative distance and the features of each vehicles and the road. It was developed by Treiber, Hennecke, and Helbing in 2000.

The formula used is the following one :

$$v_{t+1} = v_t + \Delta t \cdot u_t$$

with

$$u_t = \max\left(a\Big(1 - \Big(\frac{v_t}{v_0}\Big)^{\delta} - \Big(\frac{s^*}{s}\Big)^2\Big), -B\right),$$

$$s^* = s_0 + \max\left(0, v_t T + \frac{v_t \Delta v}{2\sqrt{ab}}\right).$$

Variable	Value	Remark
v_t	/	Speed at time t
Δt	0.1 s	Time between 2 consecutive steps
u_t	/	Acceleration at time t
v_0	Depends of the road	Maximum allowed speed
s	/	Distance to the front car
Δv_t	/	Speed difference between the agent and the next car
s_0	2m	Desired minimal distance to next car
δ	4	Influence of the speed
T	1.5 s	Time headway
a	0.73 m/s^2	Maximal acceleration
b	1.67 m/s^2	Confortable breaking
B	8 m/s^2	Maximal breaking

Table 7.2: Parameters of the IDM speed calculation model

Trajectory calculus

The trajectory of each simulated vehicle is computed by following the center of its assigned lane. The lanes are described as oriented polylines, and the road to follow is first selected by an angular criterion: we choose the closest road whose direction is consistent with the vehicle's heading.

In order to ensure smooth movements, the vehicle does not directly target the closest point of the polyline (which could cause sharp and unrealistic changes of direction if the vehicle is

far from the road). Instead, the 10^{th} point on the polyline is chosen as the first reference. To further improve smoothness, we generate intermediate path points by linearly interpolating between the road points and placing a reference point every meter. The final trajectory is obtained by fitting a B-spline through these interpolated points, which the vehicle then follows continuously.

Special features of the implementation

Several specific implementation choices were necessary to ensure a stable and realistic simulation:

- Only the vehicles valid at both time 9 and 10 are considered in the simulation. This guarantees that each vehicle has an initial position and speed, which are required inputs to the IDM formulation.
- A maximal braking parameter B was added to the acceleration law. Without this constraint, the IDM may diverge in extreme situations, leading to unrealistic accelerations.
- To simulate the effect of a red traffic light, we introduce an invisible vehicle fixed at the stop line. Other vehicles react to it as if it were a regular obstacle.
- Vehicles that are not actively simulated (for example, because they appear or disappear in the dataset) are removed from the scenario. Otherwise, their irregular trajectories could mislead the IDM and generate artifacts.

Default of the method

Although IDM is widely used and robust, several limitations must be mentioned in the context of this implementation:

- The model requires a precise selection of the leading vehicle. Incorrect selection can produce unrealistic behaviors.
- Parked vehicles or static obstacles are not modeled, which may lead to overly optimistic trajectories in complex urban settings.
- The lane-following approach based on B-spline interpolation, while effective, may produce trajectories that appear mechanical and lack the variability of human driving.

Despite these defaults, the method remains suitable for rule-based scenario generation, where interpretability and reproducibility are prioritized over perfect human-likeness.

Computing details

The simulation runs at a time step of $\Delta t = 0.1$ seconds. At each step, the acceleration is computed using the IDM formula, updated speed is obtained, and positions are advanced along the precomputed trajectory spline. Only vehicles with valid initialization (positions and velocities at time 9 and 10) are included in the computation. The system is therefore efficient, as it avoids handling inconsistent or partial trajectories.

7.2 SMART Details

SMART implementation

SMART (Scenario Modeling with Attention-based Relational Transformers) is a deep-learning-based trajectory prediction model for the WOSAC challenge. It follows a structured encoder-decoder architecture where agents and map features are jointly embedded and processed through relational attention layers. In this appendix, we present the structure of the original model and then detail the modifications we introduced for conditioning with latent variables.

Original architecture

The original SMART implementation is composed of three main components:

- **SMART Decoder**: acts as the central decoder, combining a map encoder and an agent encoder.
- **SMART MapDecoder**: embeds the static map elements, represented as polylines, and computes lane-to-lane (pl2pl) relational embeddings.
- **SMART AgentDecoder**: processes dynamic agents (vehicles, pedestrians, cyclists) using temporal, agent-to-agent (a2a), and map-to-agent (pl2a) attention layers.

The pipeline is as follows:

1. Each agent is tokenized into a sequence of motion tokens (positions, headings, type, and shape), which are embedded using a combination of Fourier and MLP embeddings.
2. Relational edges are built between:
    - **Temporal edges**: connecting the same agent across different timesteps.
    - **Agent-to-agent edges**: encoding interactions within a fixed radius.
    - **Map-to-agent edges**: linking each agent with nearby lane points.
3. Attention layers (temporal, pl2a, and a2a) propagate information across these graphs.
4. A final MLP head predicts the next motion token logits, enabling trajectory sampling either in open-loop (direct prediction) or closed-loop (autoregressive rollout) fashion.

The model outputs both token-level predictions (for classification of motion primitives) and trajectory-level predictions (position and heading over future timesteps). During inference, rollouts are generated by iteratively sampling next-step tokens and updating the agent state.

Modified architecture with latent conditioning

To integrate higher-level scenario control, we modified the **SMARTAgentDecoder** to condition agent embeddings on latent vectors extracted from a Variational Autoencoder (VAE). The changes are as follows:

- A new module `latent_to_condition` (two-layer MLP) maps the latent vector $z \in R^{56}$ to the hidden dimension of the decoder.

- The conditioning vector is added to the fused agent features before entering the attention layers:
$$\tilde{f}_a = f_a + W_2\,\sigma(W_1 z),$$
where f_a is the original fused embedding, z is the latent vector, and σ is a ReLU activation.
- Both the `forward` and `inference` passes of the agent decoder now accept an optional `latent_vec` argument.

This modification allows the decoder to bias its trajectory predictions based on external latent variables, which in our implementation are derived from the acceleration profiles of agents.

Transfer learning setup

The integration of latent conditioning required adjustments to the training setup:

- A pretrained VAE encodes agent acceleration time-series into latent vectors z. The VAE is loaded in evaluation mode, and its weights are frozen.
- During SMART training, only two modules are finetuned:
  - The `latent_to_condition` layers, which map z into the agent feature space.
  - The `fusion_emb` layer, ensuring proper integration between latent conditioning and motion embeddings.
- All other SMART parameters remain frozen, effectively turning the procedure into a transfer learning scheme where the pretrained trajectory model is adapted to use external conditioning.

Summary of contributions

Compared to the original SMART implementation, our modifications:

1. Extend the agent decoder with latent conditioning capabilities.
2. Introduce a transfer learning pipeline where only the conditioning layers are trained, leveraging pretrained SMART weights.
3. Enable scenario generation that can be controlled by high-level latent variables derived from VAE-encoded agent behaviors.

(See tab 7.3 and tab 7.4)

7.3 Dataset

7.3.1 Dataset details

The training dataset consists of 490,000 scenarios, while the validation dataset includes 44,100 scenarios. Both datasets are originally stored in TensorFlow TFRecord files.

Category	Value	Explanation
Embedding size	128	Dimension of all feature embeddings and attention layers.
Fourier features	64	Number of frequency bands for continuous inputs.
Attention heads	8 (16 per head)	Multi-head attention setup.
Dropout	0.1	Regularization to prevent overfitting.
History drop prob.	0.1	Randomly removes history points during training for robustness.
Map layers	3	Depth of attention layers for the map encoder.
Agent layers	6	Depth of attention layers for the agent encoder.
Map–map radius	10 m	Neighborhood size for map connectivity.
Map–agent radius	30 m	Distance within which lanes influence agents.
Agent–agent radius	60 m	Distance for interaction between agents.
Temporal span	30 steps	Maximum history length used in temporal edges.
History window	11 steps	Number of past steps (at 10 Hz).
Prediction horizon	80 steps	Future trajectory length (at 10 Hz).

Table 7.3: Main SMART hyperparameters for architecture, graph structure, temporal settings, and training.

Category	Value	Explanation
Input dimension	1	Single feature per timestep (acceleration).
Hidden size	256	Width of the VAE encoder/decoder.
Layers	2	Depth of the VAE MLP.
Latent dimension	56	Size of the latent vector used to condition SMART.
Integration	$56 \rightarrow 128$	Latent vector is projected to the SMART embedding size.
Trainable parts	Conditioning layers only	All pretrained SMART weights are frozen except for latent conditioning and fusion.

Table 7.4: VAE hyperparameters and integration into SMART through latent conditioning.

Key	Description	Tensor Dimensions	Data Type & Value Range
roadgraph_samples/xyz	Coordinate positions of sampled map data points (0.5m sampling).	[num_points, 3]	float — Global coords (m)
roadgraph_samples/dir	Unit direction vector for each map feature sample point.	[num_points, 3]	float — Unit vectors
roadgraph_samples/type	Unique int for each map feature type & properties.	[num_points, 1]	int64 — [0–19], e.g. LaneCenter-Freeway=1
roadgraph_samples/valid	Valid flag for each map sample point.	[num_points, 1]	int64 — {0,1}

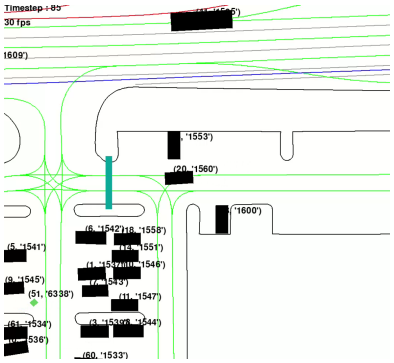
Key	Description	Tensor Dimensions	Data Type & Value Range
roadgraph_samples/id	Unique ID for the vector map feature of each sample.	[num_points, 1]	int64 — Any 64-bit int
scenario/id	Unique string ID for the scenario.	[1]	string — Hex string
state/current/x	X coordinate of each object at each time step.	[num_objects, num_steps]	float — Global coords (m), padded 1
state/current/y	Y coordinate of each object at each time step.	[num_objects, num_steps]	float — Global coords (m)
state/current/z	Z coordinate of each object at each time step.	[num_objects, num_steps]	float — Global coords (m)
state/current/bbox_yaw	Heading angle of object.	[num_objects, num_steps]	float — radians
state/current/length	Length of object.	[num_objects, num_steps]	float — meters
state/current/width	Width of object.	[num_objects, num_steps]	float — meters
state/current/height	Height of object.	[num_objects, num_steps]	float — meters
state/current/speed	Speed of object.	[num_objects, num_steps]	float — m/s
state/current/ timestamp_micros	Timestamp per step.	[num_objects, num_steps]	int64 — μs
state/current/vel_yaw	Yaw of velocity vector.	[num_objects, num_steps]	float — radians
state/current/velocity_x	X component of velocity.	[num_objects, num_steps]	float — m/s
state/current/velocity_y	Y component of velocity.	[num_objects, num_steps]	float — m/s
state/current/valid	Valid flag for state/current tensors.	[num_objects, num_steps]	int64 — {0,1}
state/future/x	X coordinate at future time steps.	[num_objects, num_steps]	float — Global coords (m)
state/future/y	Y coordinate at future time steps.	[num_objects, num_steps]	float — Global coords (m)
state/future/z	Z coordinate at future time steps.	[num_objects, num_steps]	float — Global coords (m)
state/future/bbox_yaw	Heading angle.	[num_objects, num_steps]	float — radians
state/future/length	Length of object.	[num_objects, num_steps]	float — m
state/future/width	Width of object.	[num_objects, num_steps]	float — m
state/future/height	Height of object.	[num_objects, num_steps]	float — m
state/future/speed	Speed of object.	[num_objects, num_steps]	float — m/s

Key	Description	Tensor Dimensions	Data Type & Value Range
state/future/ timestamp_micros	Timestamps.	[num_objects, num_steps]	int64 — μs
state/future/vel_yaw	Yaw of velocity vector.	[num_objects, num_steps]	float — radians
state/future/velocity_x	X velocity component.	[num_objects, num_steps]	float — m/s
state/future/velocity_y	Y velocity component.	[num_objects, num_steps]	float — m/s
state/future/valid	Valid flag for future tensors.	[num_objects, num_steps]	int64 — {0,1}
state/past/x	X coordinate at past time steps.	[num_objects, num_steps]	float — Global coords (m)
state/past/y	Y coordinate at past time steps.	[num_objects, num_steps]	float — Global coords (m)
state/past/z	Z coordinate at past time steps.	[num_objects, num_steps]	float — Global coords (m)
state/past/bbox_yaw	Heading angle.	[num_objects, num_steps]	float — radians
state/past/length	Length.	[num_objects, num_steps]	float — m
state/past/width	Width.	[num_objects, num_steps]	float — m
state/past/height	Height.	[num_objects, num_steps]	float — m
state/past/speed	Speed.	[num_objects, num_steps]	float — m/s
state/past/timestamp_micros	Timestamps.	[num_objects, num_steps]	int64 — μs
state/past/vel_yaw	Yaw of velocity vector.	[num_objects, num_steps]	float — radians
state/past/velocity_x	X velocity component.	[num_objects, num_steps]	float — m/s
state/past/velocity_y	Y velocity component.	[num_objects, num_steps]	float — m/s
state/past/valid	Valid flag for past tensors.	[num_objects, num_steps]	int64 — {0,1}
state/tracks_to_predict	Flags of objects to predict.	[num_objects]	int64 — Indices
state/objects_of_interest	Flags of objects with interactive behavior.	[num_objects]	int64 — {0,1}
state/id	Object ID.	[num_objects]	float — ID
state/is_sdc	Flag if object is the AV.	[num_objects]	int64 — {0,1}
state/type	Type of object.	[num_objects]	float — Vehicle=1, Pedestrian=2, Cyclist=3
traffic_light_state/ current/state	Traffic light states.	[num_steps, num_light_positions]	int64 — 0–8 codes
traffic_light_state/ current/x	X pos of light.	[num_steps, num_light_positions]	float — Global coords (m)

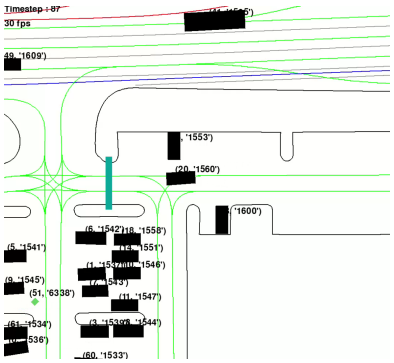
Key	Description	Tensor Dimensions	Data Type & Value Range
traffic_light_state/ current/y	Y pos of light.	[num_steps, num_light_positions]	float — Global coords (m)
traffic_light_state/ current/z	Z pos of light.	[num_steps, num_light_positions]	float — Global coords (m)
traffic_light_state/ current/id	Lane ID controlled.	[num_steps, num_light_positions]	int64
traffic_light_state/ current/valid	Valid flag.	[num_steps, num_light_positions]	int64 — {0,1}
traffic_light_state/ current/timestamp_micros	Timestamp.	[num_steps]	int64 — μs
traffic_light_state/ future/state	Traffic light states (future).	[num_steps, num_light_positions]	int64 — 0–8 codes
traffic_light_state/ future/x	X pos.	[num_steps, num_light_positions]	float — Global coords (m)
traffic_light_state/ future/y	Y pos.	[num_steps, num_light_positions]	float — Global coords (m)
traffic_light_state/ future/z	Z pos.	[num_steps, num_light_positions]	float — Global coords (m)
traffic_light_state/ future/id	Lane ID.	[num_steps, num_light_positions]	int64
traffic_light_state/ future/valid	Valid flag.	[num_steps, num_light_positions]	int64 — {0,1}
traffic_light_state/ future/timestamp_micros	Timestamps.	[num_steps]	int64 — μs
traffic_light_state/ past/state	Traffic light states (past).	[num_steps, num_light_positions]	int64 — 0–8 codes
traffic_light_state/ past/x	X pos.	[num_steps, num_light_positions]	float — Global coords (m)
traffic_light_state/ past/y	Y pos.	[num_steps, num_light_positions]	float — Global coords (m)
traffic_light_state/ past/z	Z pos.	[num_steps, num_light_positions]	float — Global coords (m)
traffic_light_state/ past/id	Lane ID.	[num_steps, num_light_positions]	int64
traffic_light_state/ past/valid	Valid flag.	[num_steps, num_light_positions]	int64 — {0,1}
traffic_light_state/ past/timestamp_micros	Timestamps.	[num_steps]	int64 — μs

7.3.2 Dataset anomalies

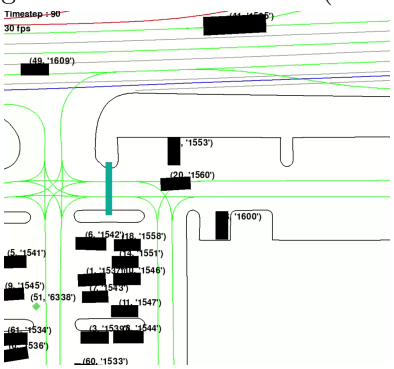
Impact of small fluctuations on the acceleration



(a) The vehicle 1560 is at a given position



(b) The vehicle 1560 makes a slight movement backward (to the right) due to a detection error



(c) The vehicle 1560 is replaced at the right spot

Figure 7.1: Illustration of a perception error where a vehicle is momentarily placed in an inconsistent position across three successive frames. This results in a slight back-and-forth displacement being registered as an acceleration spike of approximately $10g$, highlighting the impact of noisy perception on derived motion metrics.