Informatik 11
Embedded Software

RWTH AACHEN UNIVERSITY

Master Thesis

Masterarbeit

AI-Based Generation of Testing Scenarios for Motion Planners on Connected and Automated Vehicles

KI-basierte Erzeugung von Test-Szenarien für Bewegungsplaner in vernetzten und automatisierten Fahrzeugen

Ruizhang Zhou
Matrikelnummer: 405527

Aachen, April 29, 2024

Examiners:
Prof. Dr.-Ing. Stefan Kowalewski
Prof. Dr.-Ing. Lutz Eckstein

Advisors:
M.Sc. Jianye Xu
M.Sc. Armin Mokhtarian

This thesis was submitted to
Lehrstuhl Informatik 11 – Embedded Software

Acknowledgments

Ex ante, I'd like to thank the listed persons below who supported me during the editing time of this thesis:

- Jianye Xu, Armin Mokhtarian for supporting.
- Simon Schäfer's work for Data Converter and CPM Remote

This is to be done

Abstract

In the rapidly evolving field of automated vehicles, the development of reliable motion planning systems is critical. These systems must effectively navigate complex real-world scenarios. Traditional datasets, while useful, often lack the variability and specificity needed to robustly train these systems. This thesis explores the use of artificial intelligence (AI) to generate synthetic testing scenarios that could potentially bridge this gap, enhancing the robustness of motion planners through comprehensive and diverse training datasets.

Eidesstattliche Versicherung

Zhou, Ruizhang	405527
Name, Vorname	Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Masterarbeit mit dem Titel

KI-basierte Erzeugung von Test-Szenarien für Bewegungsplaner in vernetzten und automatisierten Fahrzeugen

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Ort, Datum	Unterschrift

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Ort, Datum	Unterschrift

Contents

List of Tables

List of Figures

1. Introduction

The rapid advancement in connected and automated vehicle technologies has underscored the critical need for robust motion planning systems that ensure safety and efficiency on the road. However, developing these systems requires extensive testing under diverse scenarios, many of which may not be readily available or may occur too infrequently in the real world to gather substantial data. My motivation of this thesis is that, data comes from real world, and is hard to get and sometimes not open. One workaround for this difficultly would be to create sufficiently realistic synthetic data. This synthetic data could then be used to reproducibly develop and train machine learning models, enabling better experiments, and ultimately better models for connected autonomous vehicles.

This thesis addresses the challenge by employing artificial intelligence to generate synthetic testing scenarios that are both realistic and sufficiently varied to mimic complex real-world conditions. This approach not only enhances the development and testing phases but also provides a scalable solution to generate data that are difficult, expensive, or unethical to collect in reality.

1.0.1. Motivation

The impetus for exploring synthetic data generation stems from the intrinsic challenges associated with sourcing real-world data. Real-world data, particularly in the domain of connected autonomous vehicles, is notoriously difficult to acquire due to several impediments: it often requires complex setups, expensive hardware, and specific licenses. Moreover, such data is rarely open or accessible, limiting the scope of research and development.

The availability of scenario collections that capture real-world conditions is equally constrained. These collections are typically bespoke to particular environments and specific entities such as pedestrians, bicycles, cars, and buses. This specificity restricts their applicability across the diverse scenarios that autonomous vehicles must navigate.

Additionally, the process of extracting valuable insights from real-world data is marked by inefficiency. Often, only small segments of data, perhaps 10 minutes from 10 hours of recording, are genuinely challenging or interesting. This necessitates a labor-intensive manual selection process, which not only is time-consuming but also introduces a bias towards more dramatic, less frequently occurring scenarios.

Given these constraints, synthetic data emerges as a potent alternative. By leveraging advanced simulation technologies, it is possible to create highly realistic and diverse data sets that mimic complex real-world scenarios. This approach not only circumvents the logistical and financial barriers associated with real data collection but also offers a reproducible and scalable method to train and refine machine learning models. Ultimately, the use of synthetic data can lead to more robust experimentation and the development of superior models, accelerating the advancement of connected autonomous vehicle technologies.

1.1. Main contributions

This document aims to introduce a structured template and methodology for generating test scenarios using advanced machine learning models, particularly Generative Adversarial Networks (GANs) and Denoising Diffusion Probabilistic Models, tailored for the motion planning algorithms in connected and automated vehicles. The primary contributions are:

1. Developing an AI-based framework to preprocess real-world data into time series test cases suitable for training machine learning models.
2. Utilizing cutting-edge generative models to produce diverse and representative synthetic datasets.
3. Integrating these datasets into the CPM Remote system, facilitating seamless simulation and evaluation workflows.

1.2. Outline

The structure of this thesis is designed to guide the reader through the process of scenario generation, from theoretical foundations to practical applications:

- **Chapter 2** provides a foundational overview of the necessary theoretical concepts, including a review of relevant literature in the field of AI-generated testing environments.
- **Chapter 3** discusses previous works and how they relate to the current thesis, highlighting the advancements made by this research.
- **Chapter 4** details the methodologies employed for generating synthetic data using AI models, focusing on the algorithms, model architecture, and the rationale behind choosing specific approaches.
- **Chapter 5** describes the practical implementation of the methodologies discussed in the previous chapter, including software tools used, data preprocessing

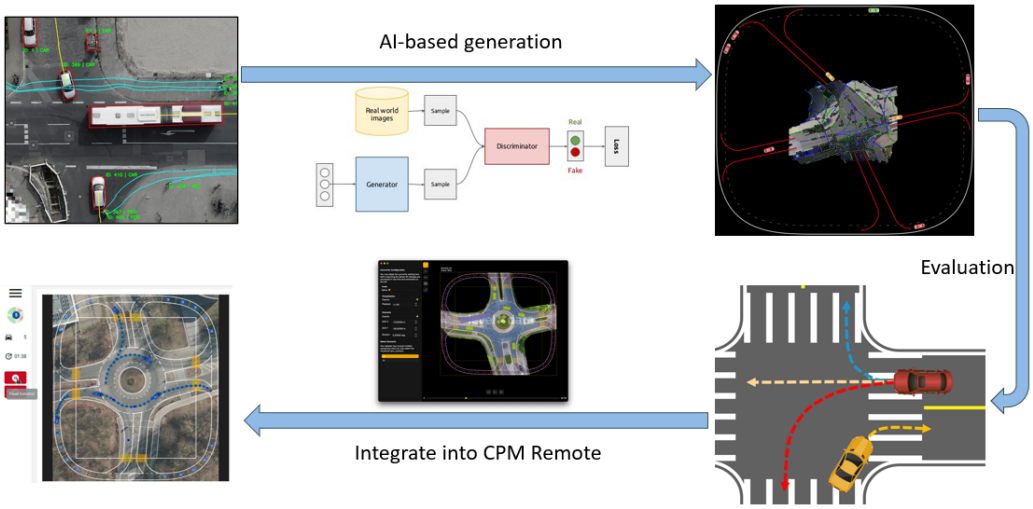


Figure 1.1.: Overview of the thesis

techniques, and the step-by-step process to integrate the generated data into the CPM Remote system.

- **Chapter 6** evaluates the synthetic data against real-world data to assess their quality and utility in testing motion planners.
- **Chapter 7** concludes the thesis with a summary of findings and potential areas for future research.

By the end of this thesis, the reader will appreciate the complexities involved in creating high-fidelity test scenarios and the potential of AI to revolutionize testing methodologies in the automotive industry.

2. Background

This background provides the necessary theoretical underpinnings and technical groundwork essential for appreciating the novel contributions in synthetic data generation for motion planning in autonomous vehicles discussed in subsequent chapters. The focus is firmly on leveraging synthetic data to enhance the training of algorithms responsible for the safe and effective operation of autonomous vehicles in complex environments.

The integration of AI into testing scenarios for connected and automated vehicles is predicated on the capability of AI models to generate data that mimics real-world conditions. This involves understanding the dynamics of vehicle movement and interactions within varied traffic scenarios, which are often dictated by complex environmental variables. The need for synthetic data arises from several challenges associated with real-world data collection, including high costs, limited availability, and privacy concerns. Moreover, synthetic data can be tailored to include rare but critical scenarios that might not be well-represented in real-world data, providing a more comprehensive training landscape for motion planners.

2.1. Theoretical Foundations of Motion Planning

2.1.1. Motion Planning

The fundamental challenge in the development of autonomous vehicles is the design of robust motion planning systems. These systems enable vehicles to determine viable paths from a start point to a destination while navigating around obstacles safely and efficiently. This process involves prediction models that accommodate dynamic elements such as pedestrian movements, changes in traffic conditions, and interactions with other vehicles.

2.1.2. Trajectory Optimization

Trajectory planning is at the heart of motion planning, where the aim is to find the optimal path that minimizes travel time and energy consumption while maximizing safety. This involves solving complex optimization problems that take into account vehicle dynamics, road conditions, and traffic laws.

2.2. Role of Synthetic Data in Motion Planning

2.2.1. Challenges with Real-World Data

While real-world data is invaluable for training motion planning algorithms, it poses significant challenges including:

- **High Costs and Limited Accessibility:** Gathering comprehensive driving data involves extensive time, effort, and resources.
- **Privacy Issues:** Data collected from public areas may contain personally identifiable information, which must be handled with strict compliance to privacy laws.
- **Underrepresentation of Rare Events:** Critical scenarios such as near-accidents or unusual weather conditions are rare in collected data but crucial for training robust models.

2.2.2. Benefits of Synthetic Data

Synthetic data generation via generative models offers a promising alternative by enabling the simulation of diverse driving scenarios which might not be readily available in the real-world datasets.

- **Control Over Scenarios** It allows for the creation of specific conditions that are either rare or non-existent in the available real-world data.
- **Cost Effectiveness** Reduces the reliance on expensive real-world data acquisition campaigns.
- **Privacy Compliance** Eliminates the risk of privacy breaches often associated with real-world data.

2.2.3. Data types of Generative models

The challenge of generating synthetic time series data using a GAN is greater than generating synthetic images using a GAN. In addition to learning the distribution of each given point (e.g., the distribution of stock prices at a given timestamp), GANs need to learn the temporal dynamics - the dynamics that drive the correlation between two series.
That is, in addition to learning the distribution of individual features under a given point in time, it also needs to learn the relationship between individual features. Sequence correlation, as an intrinsic property of time series data, poses a big challenge to generative models that aim to generate time series. A good generative model

should not only learn the distribution of features within each time point, but also learn the potentially complex dynamics of those variables across time). The approach of RGANs is to model the distribution of features at each time point, and accordingly, their loss is simply summed up at each time point, which is an insufficient training method to capture serial correlation.The difference between RGANs and images is that the size is not fixed (the number of cars in each case is different, and the frame in which each car appears in the case is also different).

2.2.4. Generative Models in Data Synthesis

- **Generative Adversarial Networks (GANs)** These consist of two networks, the generator and the discriminator, which work in tandem to produce data indistinguishable from real-world data as perceived by the discriminator.

- **Variational Autoencoders (VAEs)** VAEs are powerful generative models that learn the latent spaces of high-dimensional data, allowing them to generate new data points with variations controlled via the latent space.

- **Conditional and Time-Series GANs** These models extend GANs by conditioning the data generation process on additional labels (conditional GANs) or by generating data across different time steps (TimeGANs), which is crucial for sequential prediction tasks like motion planning.

- **Diffusion Models**: Diffusion models are a class of generative models that transform a simple distribution (e.g., Gaussian noise) into a complex data distribution through a learned gradual denoising process. They are particularly noted for generating high-quality images and are being explored for their potential in generating realistic sequential data for applications like motion planning.

- **Diffusion-TS**: An extension of basic diffusion models, Diffusion-TS (Diffusion for Time Series) specifically adapts the diffusion process for time-series data. It models the data generation as a sequence of conditional distributions, making it suitable for generating realistic vehicle trajectories in motion planning scenarios.

2.2.5. Evaluation Metrics for Synthetic Data

- **Inception Score (IS) and Fréchet Inception Distance (FID):** These metrics are used to assess the quality and diversity of generated images in the context of GANs and are adapted here for evaluating synthetic trajectories in terms of realism and variety.

- **t-Distributed Stochastic Neighbor Embedding (t-SNE):** A machine learning algorithm used to visualize high-dimensional data by reducing it to two or three dimensions, thus facilitating easier inspection of the data's characteristics.

2.3. Utilized Notations and Diagram Types

The thesis employs standard notations where X denotes input features, Y denotes outputs, Z represents latent variables or noise used in data generation, and T encapsulates temporal aspects relevant to the discussion. Various diagrams, such as flowcharts and UML diagrams, are utilized to illustrate systems' architectures and the flow of data.

3. Related Work

This chapter outlines the foundational works and datasets that underpin the methodologies employed in this thesis. It discusses a variety of generative models, especially those relevant to synthetic data generation for autonomous driving scenarios. The datasets used in this study, namely inD and rounD, are introduced to highlight their application within the broader context of this research.

3.1. Motion Planners of Connected and Automated Vehicles

Motion planning is a crucial aspect of connected and automated vehicles (CAVs), aimed at generating safe, feasible, and efficient paths for vehicles within dynamic environments. Traditional graph-based planners like Dijkstra's algorithm and A* utilize a discretized map to plot paths, although they struggle with high-dimensional data and complex scenarios. This limitation is addressed by sampling-based planners such as Rapidly-exploring Random Trees (RRT) and Probabilistic Roadmaps (PRM), which randomly sample the environment and build a path framework that vehicles can follow. Recently, machine learning has been pivotal in motion planning, providing data-driven insights and predictions that enhance the planner's contextual awareness. Techniques like reinforcement learning and imitation learning allow planners to learn from simulated environments and human driving behaviors, respectively, fostering policies that promote safety, compliance, and passenger comfort.

3.2. Generative Models for Synthetic Data Generation in Traffic Systems

Generative models have revolutionized the field of synthetic data generation, offering powerful tools for creating realistic and diverse datasets necessary for training machine learning models, especially in situations where real data may be limited or privacy-sensitive. These models are particularly pivotal in developing autonomous vehicle technologies, where they generate dynamic traffic scenarios to ensure robust testing of motion planning algorithms.

Generative Adversarial Networks (GANs) are comprised of two competing networks: a generator that creates samples intended to mimic the real data distribution, and

a discriminator that attempts to distinguish genuine data from synthesized ones. This model has been widely successful in generating realistic image datasets and has recently been adapted to handle sequential data such as traffic patterns through TimeGANs, which modify the traditional GAN architecture to generate sequential, time-series data that is temporally coherent.

Generative Adversarial Networks (GANs) have been extended beyond their initial application in static image generation to address the complexities of sequential, time-sensitive data. Various adaptations of the original GAN architecture have been developed to generate time-series data that is temporally coherent and contextually relevant. These variants include:

- **TimeGAN**: Specifically designed to address the temporal coherence in generated sequences, TimeGAN integrates supervised loss into the traditional adversarial framework. This ensures that the generated sequences not only mimic the real data distribution but also adhere to the temporal dynamics inherent in the data. TimeGAN has been applied successfully in domains requiring realistic simulation of sequential data, such as financial forecasting and medical data analysis.
- **C-RNN-GAN (Conditional Recurrent Neural Network GAN)**: This variant utilizes conditioned recurrent neural networks in both the generator and discriminator to handle the additional context that influences the sequence generation. C-RNN-GAN is particularly useful in scenarios where the sequence output needs to be conditioned on external factors, such as environmental changes affecting traffic or market conditions.
- **SeqGAN**: SeqGAN treats sequence generation as a reinforcement learning problem where the generator improves its sequence output based on feedback from the discriminator, treated as a reward signal. This model is especially effective for generating textual data and can be adapted for any sequential data generation tasks.
- **SAGAN (Sinusoidal Activation Generative Adversarial Networks)**: Employing sinusoidal activation functions, SAGAN exploits the natural periodicity of these functions to better model cyclic patterns in time-series data. It is well-suited for generating audio sequences and other types of periodic signals.
- **T-CGAN (Temporal Convolutional GAN)**: Incorporating temporal convolutional layers, T-CGAN is designed to handle sequences with complex temporal structures efficiently, reducing the necessity for extensive recurrence and thereby simplifying the model and enhancing training stability. Applications include video frame prediction and complex activity recognition where understanding longer temporal dependencies is crucial.

These GAN variants significantly enhance the ability to generate realistic and contextually appropriate time-series data, paving the way for innovative applications in various fields where temporal data plays a critical role.

Variational Autoencoders (VAEs) provide another method for data generation through the use of probabilistic latent spaces. By encoding input data into a latent distribution and subsequently decoding from this space, VAEs can produce new data instances that exhibit variations on the input data. This is particularly useful for modeling complex distributions of traffic data, where the nuances of the data are captured in the latent representations.

Denoising Diffusion Probabilistic Models represent a newer class of generative models that convert noise into high-quality samples through a gradual denoising process. These models have shown great promise in generating detailed and high-dimensional data, including intricate traffic scenes that can be used for the training of autonomous systems.

Diffusion Time-Series Models extend the diffusion model concept to time-series data, making it possible to generate realistic and sequential traffic data. These models are adept at capturing the temporal dynamics and the variability of traffic flows, making them ideal for simulations needed to test and validate autonomous vehicle behaviors under diverse conditions.

Generative models form a comprehensive suite of tools that can create vast and varied datasets, simulating numerous possible outcomes and scenarios in traffic systems. This capability is crucial not only for training robust and reliable autonomous vehicles but also for advancing our understanding of complex, dynamic systems such as urban traffic networks. Their continued development and integration promise to significantly enhance the safety and efficiency of autonomous transportation solutions.

3.3. Test Scenarios and Testbeds

The development and validation of generative models for traffic scenarios rely heavily on extensive and detailed real-world datasets. This thesis utilizes several prominent datasets detailed below.

3.3.1. The IKA Dataset

This dataset provides comprehensive trajectories of all road users at German intersections, captured via drone-based tracking. It serves as a critical resource for training models to navigate and understand complex intersection interactions. Similar to the inD dataset but focused on roundabouts, the rounD dataset includes trajectories that capture typical behaviors and interactions within roundabout contexts. It is vital for developing models that can generate plausible roundabout driving scenarios.

The utilization of large-scale trajectory datasets such as the inD and rounD datasets is pivotal for enhancing the development and validation of automated driving systems through data-driven methods. These datasets provide extensive and high-fidelity recordings of naturalistic road user behaviors, which are essential for training and testing predictive models used in autonomous driving.

The Institute of Automotive Engineering (IKA) at RWTH Aachen University has curated several pivotal datasets that are extensively used in the research and development of autonomous driving systems. These datasets—namely the inD (Intersection Drone), rounD (Roundabout Drone), exiD (Expressway Intersection Drone), and highD (Highway Drone)—provide detailed and comprehensive trajectories of road users across different traffic scenarios captured via drone-based tracking systems.

The **inD dataset** focuses on urban intersections, capturing over 11,500 road users including pedestrians, cyclists, cars, and trucks, which helps in understanding complex interactions at intersections. The **rounD dataset**, on the other hand, is specialized for roundabouts, tracking about 4,500 road users and offering insights into the typical behavioral patterns in roundabout navigation.

Extending the scope to high-speed road networks, the **exiD dataset** includes trajectory data from expressway intersections, highlighting maneuvers such as high-speed merging and diverging, while the **highD dataset** provides a detailed record of vehicle movements on highways, with over 110,000 vehicle trajectories captured. This dataset is particularly useful for studying high-speed driving behaviors and lane-changing dynamics on highways.

These datasets collectively cover a wide spectrum of driving environments from dense urban areas to high-speed expressways, making them invaluable for developing and testing autonomous vehicle technologies. They not only assist in training robust driving policies but also provide a benchmark for evaluating the performance of various motion planning and prediction algorithms under realistic, diverse, and challenging conditions. This comprehensive data coverage ensures that the systems developed using these datasets are well-rounded and capable of handling different driving scenarios, thereby advancing the safety and efficiency of autonomous driving technologies.

inD Dataset The *inD dataset* captures a comprehensive range of road user trajectories at German urban intersections using drone technology. With over 11,500 road users recorded, this dataset includes not only motor vehicles but also vulnerable road users like pedestrians and cyclists, presenting a rich tapestry of interactions that automated driving systems must navigate [**?**]. The data's precision, owing to the aerial perspective, significantly reduces occlusions and captures naturalistic road user behaviors, offering an ideal dataset for intersection-based studies.

rounD Dataset Similarly, the *rounD dataset* focuses on the dynamic and complex environments of roundabouts, featuring over 13,746 road users from diverse categories [**?**]. The dataset's strength lies in its detailed capture of interaction-dense scenarios typical of roundabouts, where the negotiation between different road users is frequent. The trajectories include detailed attributes such as headings, speeds, and accelerations, which are crucial for simulating and analyzing roundabout navigation and safety assessments.

Both datasets are instrumental for researchers focusing on automated driving, providing essential data for scenario-based testing, behavior prediction models, and overall traffic analysis. Their contribution to understanding detailed road user interactions in specific traffic environments (intersections and roundabouts) is invaluable for advancing autonomous vehicle technologies.

3.3.2. CPM Remote

The synthetic scenarios generated from the models are integrated into the CPM Remote framework to simulate and evaluate connected and automated vehicle technologies. This integration tests the robustness of motion planning algorithms across varied traffic conditions and contributes to safer and more efficient automated vehicle operations.

3.3.2.1. CPM Remote Competition

The CPM Remote competition stands as a cornerstone event for benchmarking the capabilities of motion planning algorithms in complex driving scenarios. It facilitates a unique platform for participants to implement and test their planning strategies using realistic simulation environments that mirror everyday driving conditions.

Competition Structure Organized by leading researchers in the field, the competition challenges are designed around the critical tasks of navigation and obstacle avoidance in dynamically changing environments. Participants are provided with a simulated vehicle equipped with standard automotive sensors and are required to navigate through a series of predefined routes involving various urban and rural scenes.

Significance The CPM Remote competition not only encourages innovation in path planning and control strategies but also emphasizes the importance of safety and efficiency. By offering a standardized testing framework, it allows for the direct comparison of different algorithms under identical conditions, thereby promoting transparency and incremental advancements in the field. The insights gained from

these challenges are expected to drive forward the development of more robust and adaptable autonomous driving technologies.

The detailed evaluations and benchmarks provided through the CPM Remote competition are essential for researchers and engineers to validate their approaches against a versatile set of challenges and peer submissions, which is critical for the iterative improvement of autonomous navigation systems.

4. Methodology

4.1. Generative Adversarial Network (GAN)

Generative Adversarial Networks (GANs), introduced by Ian Goodfellow et al. in 2014, are a revolutionary approach in unsupervised machine learning, featuring two neural networks-the generator and the discriminator-competing in a game-theoretic scenario. GANs have been widely applied in various fields such as photo-realistic image generation, art creation, and enhancing astronomical images, as well as for generating time series data which simulates realistic automotive driving scenarios.

4.1.1. Basic Principle and Structure

A GAN consists of two main components:

- **Generator (G)**: Aims to capture the data distribution, and generates new data points.
- **Discriminator (D)**: Estimates the probability that a given data point came from the training data rather than the generator.

The training procedure for GANs involves alternating between training the discriminator to distinguish real data from fake, and training the generator to fool the discriminator. The process can be described by the following minimax game:

$$\min_G \max_D \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]$$

Where:

- $\mathbf{x}$ - A real data instance from the data generating distribution p_{data}.
- $\mathbf{z}$ - A noise sample from a predefined noise distribution $p_{\mathbf{z}}$ (typically a Gaussian distribution).
- $G(\mathbf{z})$ - Data generated by the generator.
- $D(\mathbf{x})$ - The discriminator's estimate of the probability that $\mathbf{x}$ is real.

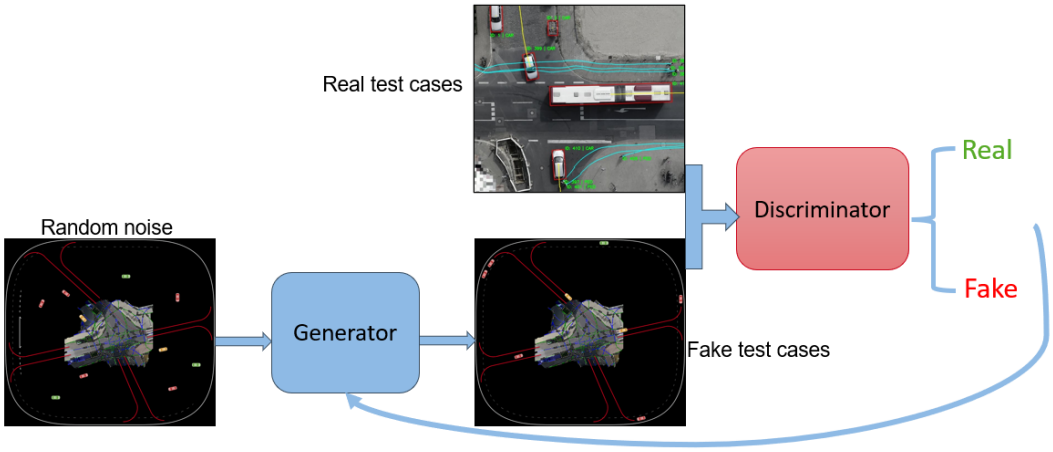


Figure 4.1.: Generative Adversarial Network (GAN)

4.1.2. Application to Vehicle Time Series Data

Originally designed for image generation, Generative Adversarial Networks (GANs) need to be carefully adapted to handle time series data due to fundamental differences in data structure. Unlike images, which are typically represented in fixed spatial dimensions, time series data involves sequences that vary over time, capturing dynamic changes. To effectively generate realistic time series data for applications such as autonomous driving technologies, GANs must be modified to account for the temporal dependencies and continuity inherent in sensor data streams from vehicles. This adaptation allows GANs to synthesize vehicle sensor data sequences that are indistinguishable from actual real-world data, providing a robust tool for testing and developing advanced driving systems.

Figure 4.2.: Image vs Time Series

4.1.2.1. Training and Challenges

- **Generator Structure:** Typically an RNN with LSTM units to capture time dependencies, taking a noise vector $\mathbf{z}$ and outputting a synthetic time series data.
- **Discriminator Structure:** An RNN, often with a similar architecture to the generator, that classifies sequences into "real" or "generated".

Challenges include ensuring temporal coherence and avoiding mode collapse, where the generator produces a limited diversity of samples. Advanced techniques such as Conditional GANs and TimeGAN have been developed to address these issues, by conditioning the generation process on additional labels or structuring the networks to better handle sequence data.

4.1.3. Loss Functions and Optimization

The objective functions for training GANs in the context of time series data involve modifications to the standard GAN loss to accommodate sequential data. The loss functions can be expressed as follows:

$$\text{Generator Loss:} \quad -\mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log D(G(\mathbf{z}))]$$

$$\text{Discriminator Loss:} \quad -\mathbb{E}_{\mathbf{x}\sim p_{\text{data}}}[\log D(\mathbf{x})] - \mathbb{E}_{\mathbf{z}\sim p_{\mathbf{z}}}[\log(1 - D(G(\mathbf{z})))]$$

These optimizations aim to strengthen the generator's ability to produce data sequences that are not only realistic but also varied and coherent over time, thereby enhancing the development of robust autonomous driving systems.

4.2. Time-series GAN (TimeGAN)

Time-series Generative Adversarial Networks (TimeGAN) extend the traditional GAN architecture to effectively model and generate realistic time-series data. The framework integrates supervised and unsupervised learning techniques to capture both the underlying temporal dynamics and the complex distribution characteristics of sequential data.

4.2.1. Architecture of TimeGAN

TimeGAN is architectured around an autoencoder for dimensionality reduction and feature learning, combined with a GAN setup for adversarial training. This dual approach ensures the generation of new, synthetic time-series data that maintains statistical and temporal integrity relative to the original data.

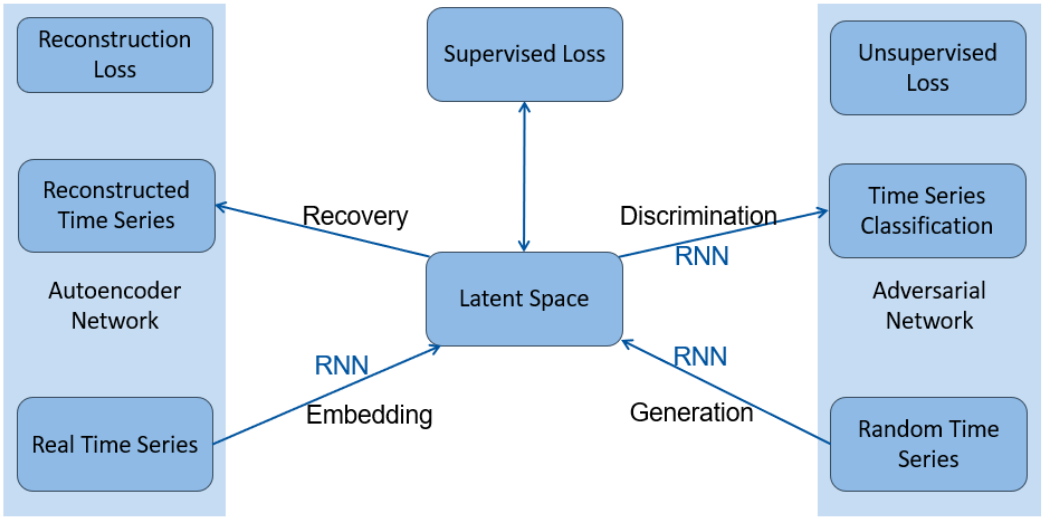


Figure 4.3.: Time-series GAN (TimeGAN)

4.2.1.1. Autoencoder Component

The autoencoder in TimeGAN is split into two networks:

- **Embedding Network (Encoder):** This network compresses the high-dimensional time-series data into a lower-dimensional latent space. The embedding is designed to retain crucial temporal information that is pivotal for the reconstruction and generation processes.

  Latent space vector $z_t = E(x_t)$, where x_t is the input time series at time t.

- **Recovery Network (Decoder):** It reconstructs the data from the latent space to its original dimensionality, aiming to minimize the reconstruction error, thus preserving the essential characteristics of the time-series data.

  Reconstructed data $\hat{x}_t = R(z_t)$, aiming to minimize $||x_t - \hat{x}_t||^2$.

4.2.1.2. Adversarial Network

The adversarial component consists of the traditional GAN setup with a generator and a discriminator:

- **Generator (G):** Takes noise vector z, possibly concatenated with latent vectors from the embedding network, to generate synthetic time-series data that mimics the real data distribution.

  $$G(z) = \text{synthetic data that mimics the true time-series data distribution.}$$

- **Discriminator (D):** Attempts to differentiate between the actual data sequences and the synthetic ones produced by the generator. This network helps in tuning the generator for better quality data production.

  $$D(x) = \text{probability that } x \text{ is a real time-series data instance.}$$

4.2.2. Training Objectives

The training of TimeGAN is conducted through a carefully orchestrated sequence of loss optimizations, detailed as follows:

1. **Reconstruction Loss:**

   $$L_{\text{Recon}} = \mathbb{E}[||x_t - \hat{x}_t||^2],$$

   focusing on minimizing the difference between original and reconstructed data, enhancing the autoencoder's fidelity.

2. **Supervised Loss:**

   $$L_{\text{Supervised}} = \mathbb{E}[||G(z_t) - x_t||^2],$$

   which fine-tunes the generator using true data instances, ensuring the synthetic data respects the time-series dynamics.

3. **Adversarial Loss:**

   $$L_{\text{GAN}} = \min_G \max_D \mathbb{E}[\log D(x)] + \mathbb{E}[\log(1 - D(G(z)))],$$

   where the generator and discriminator are trained in a min-max game to improve the realism of the generated data.

4.2.3. Significance and Applications

TimeGAN is particularly advantageous for generating time-series data that require high fidelity and temporal coherence. Its ability to synthesize realistic time-series data makes it ideal for applications in financial markets, healthcare monitoring,

and autonomous driving technologies, where precise simulation of real-world data is crucial for effective system training and evaluation.

4.3. Denoising Diffusion Probabilistic Model (Diffusion Model) as Generative Networks

Diffusion models represent a class of generative models that approach the generation process through a gradual denoising procedure. They can be viewed as a conceptual extension of Generative Adversarial Networks (GANs) where the generative process involves a more controlled and step-wise removal of noise, rather than direct generation from a random latent space.

4.3.1. Basic Concept

The fundamental concept behind diffusion models is akin to GANs in the sense that they aim to generate data that mimics real data. However, instead of the adversarial setup where a discriminator and a generator contest, diffusion models utilize a Markov chain to gradually convert a data distribution (typically Gaussian noise) into samples from the target distribution.

4.3.1.1. Process Overview

The process in diffusion models is split into two phases:

- **Forward Process (Diffusion):** This involves adding noise to the data incrementally over several steps until the data is completely converted into Gaussian noise. This process is defined by a sequence of conditional probabilities.

- **Reverse Process (Denoising):** This is the generative phase where noise is gradually removed from the noisy data to recover data points from the original data distribution. This phase essentially reverses the forward process.

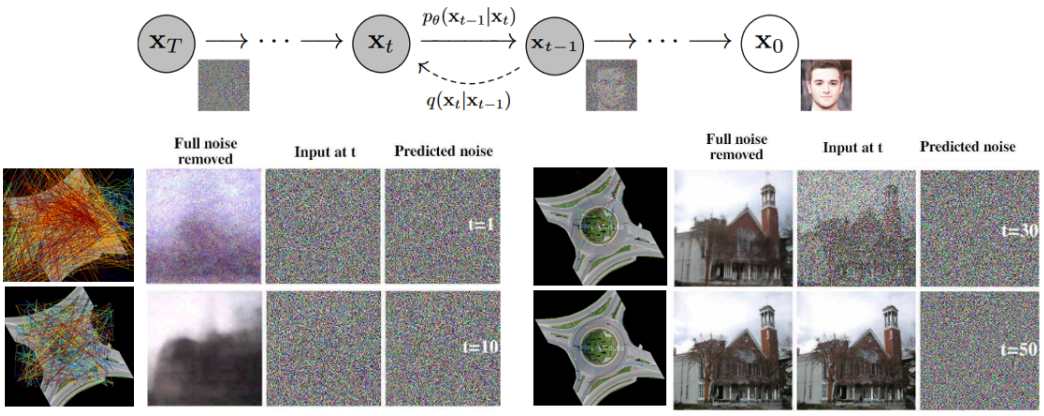


Figure 4.4.: Denoising Diffusion Probabilistic Model (Diffusion Model)

4.3.2. Connection to GANs

While traditional GANs directly learn the mapping from a noise distribution to the data distribution using a discriminator as a guide for the generator, diffusion models refine this by breaking down the mapping into simpler, incremental steps. This multistep framework allows for more stable training dynamics and the generation of higher-quality samples.

4.3.2.1. Training Dynamics

Like GANs, training diffusion models involves an optimization that minimizes a certain form of divergence (e.g., KL divergence) between the real and generated distributions. However, the process here is split across many smaller divergence minimizations per step, rather than one large minimization:

- **Generator Training:** In the context of diffusion models, the 'generator' incrementally denoises the data, conditioned on the noisy data at each step.
- **Loss Function:** The loss at each step is typically a variant of the mean-squared error between the denoised output and the data from the previous step, promoting stability.

4.3.3. Advantages Over Traditional GANs

- **Stability:** The step-wise denoising process helps mitigate the training instability commonly associated with GANs.

- **Control Over Generation:** By adjusting the noise levels and steps of the reverse process, one can control the generation process more finely than in traditional GANs.
- **Quality and Diversity:** Diffusion models are particularly noted for generating samples that are both high in quality and diversity, outperforming GANs in many benchmarks.

4.3.4. Mathematical Framework

The mathematical foundation of diffusion models is typically based on stochastic differential equations (SDEs). The forward diffusion process can be expressed as follows:

$$dX_t = \theta(X_t, t)dt + \sigma(t)dW_t$$

where X_t is the data at step t, $\theta(X_t, t)$ is the drift coefficient, $\sigma(t)$ is the diffusion coefficient, and dW_t is the increment of a Wiener process (Brownian motion).

The reverse process, which is more critical for generation, involves learning a reverse SDE conditioned on the noised data:

$$dX_t = [\theta(X_t, t) - g(t)^2 \nabla_x \log p_t(X_t)]dt + g(t)dW_t$$

where $g(t)$ is a function representing the noise level at step t, and $\nabla_x \log p_t(X_t)$ is the gradient of the log-probability of the data at step t, learned by the model.

Diffusion models, by leveraging such a controlled generative process, offer a robust and high-performing alternative to GANs, particularly for complex data distributions like those in automotive time series data.

4.4. Diffusion-TS: Theoretical Framework

Diffusion-TS leverages the denoising diffusion probabilistic model (DDPM) framework to generate multivariate time series data. This model synthesizes time series by reversing a diffusion process that gradually denoises a signal, transforming random noise into structured time series data through a learned neural network.

4.4.1. Model Architecture

The architecture of Diffusion-TS is built upon an encoder-decoder mechanism with transformers and employs a disentangled representation of temporal dynamics, which enhances interpretability and generation quality.

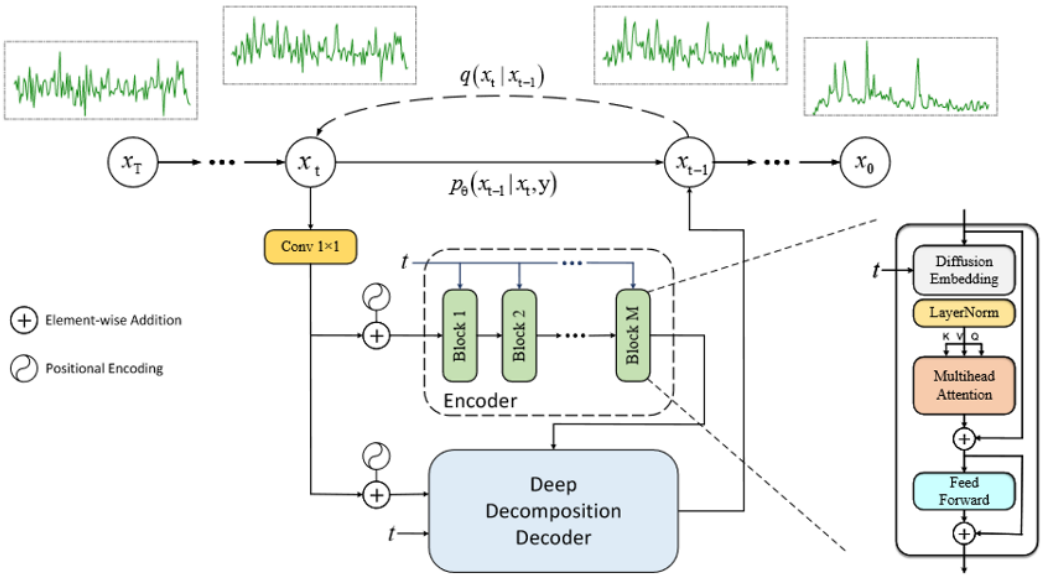


Figure 4.5.: Diffusion Time-Series (Diffusion-TS)

4.4.1.1. Encoder-Decoder Transformer

The core of the Diffusion-TS architecture is an encoder-decoder transformer that processes time series data by first encoding it into a latent space and then decoding it to generate synthetic time series:

- **Encoder:** Maps input time series into a latent representation which captures underlying temporal dynamics.

- **Decoder:** Uses the latent representation to reconstruct the time series, ensuring that essential temporal patterns are preserved and noise is reduced.

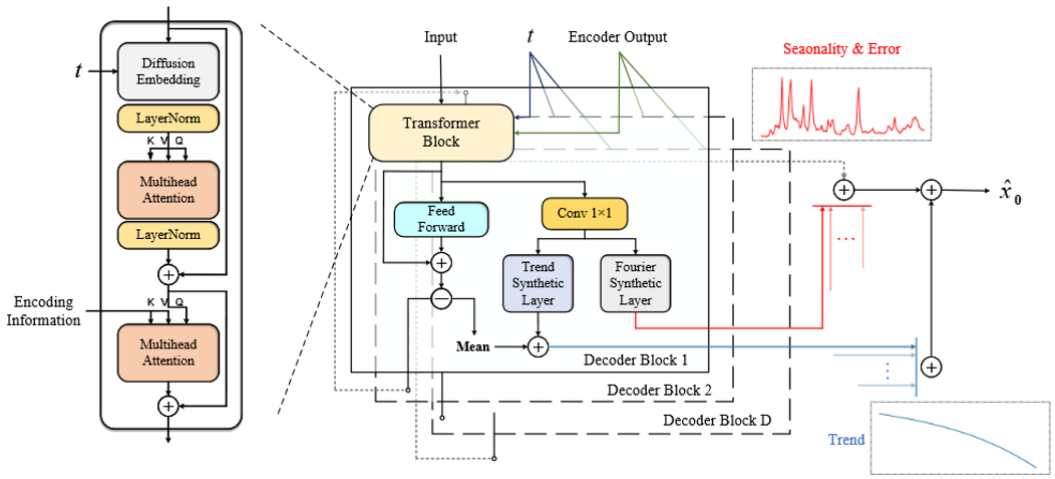


Figure 4.6.: Decoder of Diffusion Time-Series (Diffusion-TS)

4.4.1.2. Disentangled Representation

The model disentangles the time series into trend, seasonality, and residual components, allowing for targeted modeling of each aspect:

- **Trend Component (V_t):** Captures the non-periodic changes over time.
- **Seasonality Component (S_t):** Models the periodic changes, encapsulating patterns that repeat at fixed intervals.
- **Residual Component (R_t):** Accounts for the noise and any other unmodeled part of the time series.

4.4.2. Loss Function

The training of Diffusion-TS involves a Fourier-based loss function which ensures the model accurately captures the time series' characteristics through both the time and frequency domains. The loss function is defined as:

$$L(\theta) = \mathbb{E}_{t,x_0}\left[\|x_0 - \hat{x}_0(x_t, t, \theta)\|^2 + \lambda\|\mathcal{F}(x_0) - \mathcal{F}(\hat{x}_0(x_t, t, \theta))\|^2\right] \tag{4.1}$$

where:

- $\mathcal{F}$ denotes the Fourier transform,
- x_0 is the original time series,
- $\hat{x}_0(x_t, t, \theta)$ is the reconstruction from the model,

- λ is a weighting term balancing time and frequency domain reconstruction.

4.4.3. Training Dynamics

The model is trained through a reverse diffusion process starting from noise:

- **Initialization:** Begin with a Gaussian noise sequence.
- **Reverse Diffusion:** Iteratively apply the reverse model $\mu_\theta(x_t, t)$ to step back from noise to data:

$$x_{t-1} = \mu_\theta(x_t, t) + \sigma_t \epsilon, \quad \epsilon \sim \mathcal{N}(0, I) \tag{4.2}$$

  where σ_t decreases as t approaches 0.

4.4.4. Conditional Generation

Diffusion-TS can also be conditioned on additional variables (e.g., external time series) to perform tasks like forecasting and imputation:

$$x_{t-1|y} \sim \mathcal{N}(\mu_\theta(x_t, t, y), \sigma_t^2 I) \tag{4.3}$$

where y could represent external variables or past values of the time series.

4.4.5. Performance and Adaptability

Diffusion-TS demonstrates robust performance across various datasets, excelling in generating realistic, diverse, and interpretable time series data. It adapts well to different lengths and complexities of time series, making it suitable for a wide range of applications in time series analysis.

This framework not only extends the applicability of DDPMs to time series data but also opens up new possibilities for the interpretability and usability of generative models in time-sensitive domains.

5. Implementation

5.1. Data Preprocessing

Data preprocessing is a critical initial phase in the pipeline of generating synthetic datasets, particularly for training generative models applied in autonomous vehicle systems. This stage fundamentally shapes the dataset's quality and diversity, which directly influences the effectiveness of the subsequent machine learning models. Effective preprocessing not only refines the input data for better model training but also ensures that the generated outputs are realistic and useful for practical applications.

The primary aim of data preprocessing in the context of this thesis is to extract the most relevant and challenging scenarios from the raw datasets, which include the inD and rounD datasets focusing on intersection and roundabout traffic scenarios, respectively. The quality of training data is paramount, as it conditions the model's ability to learn and replicate complex dynamics involved in human-driven traffic behaviors. This chapter details the strategies employed to optimize data preprocessing, ensuring the synthesis of high-quality and diverse datasets that support the development and testing of motion planning algorithms in connected and automated vehicles.

Data preprocessing involves several strategic decisions and operations that collectively enhance the input data's suitability for generating synthetic time-series datasets. These operations include selecting appropriate sizes for test cases, choosing relevant features, determining the sliding window size for case extraction, and defining criteria for selecting challenging and informative cases. Each of these aspects is carefully considered to maximize the diversity and representativeness of the scenarios included in the training set, which is crucial for the robustness of generative models.

5.1.1. Resource Data from Real World

The primary datasets utilized for this research are the inD (Intersection Drone) and rounD (Roundabout Drone) datasets. Both datasets are integral to developing a comprehensive understanding and generating synthetic data reflective of real-world driving behaviors in urban intersections and roundabouts, respectively.

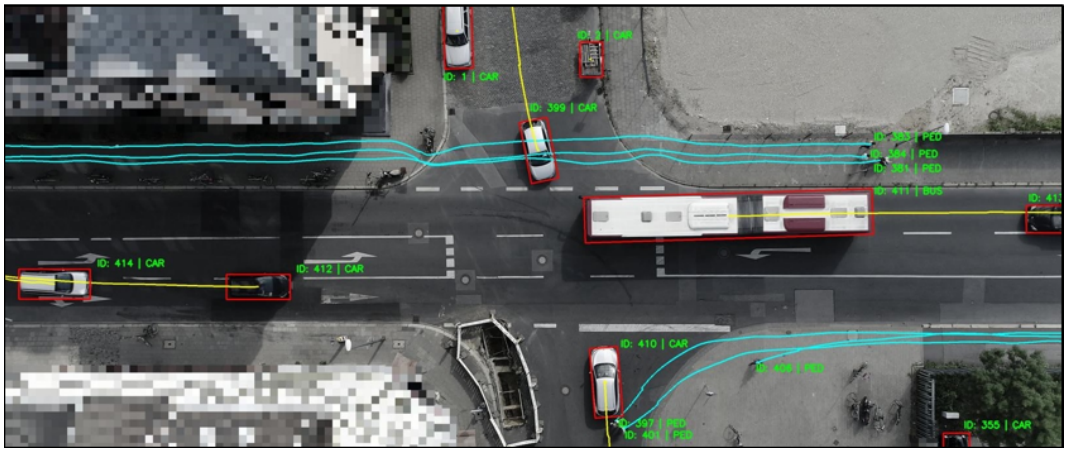


Figure 5.1.: inD dataset

Figure 5.2.: rounD dataset

5.1.1.1. Selection Rationale

- **inD and rounD Datasets:**
  - **Complex Interactions:** Both datasets encapsulate complex vehicular interactions that are critical for training robust models. The inD dataset provides extensive data on urban intersection driving, involving various dynamic elements such as pedestrians, cyclists, and cars. Similarly, the rounD dataset offers in-depth insights into the behavior of drivers within roundabouts, a critical area for autonomous navigation systems due to their circular layout and merging lanes.

  - **Diverse Scenarios:** These datasets include a wide range of driving behaviors and critical scenarios which are not typically covered in standard driving datasets. This diversity makes them ideal for training generative models to produce realistic and varied synthetic datasets.

- **Exclusion of exiD and highD Datasets:**
  - **Simplistic Traffic Patterns:** The exiD and highD datasets predominantly cover highway scenarios with vehicles mostly engaged in straightforward driving maneuvers such as lane keeping and lane changing. These scenarios lack the complexity of urban driving and do not sufficiently challenge the models' capability to handle interactive and unpredictable environments.
  - **Limited Interaction Diversity:** These datasets do not adequately represent interactive driving scenarios involving pedestrians and cyclists, which are crucial for testing autonomous vehicles in pedestrian-rich urban settings.

- **Unsuitability of uniD Dataset:**
  - **Non-standard Road Geometries:** The uniD dataset contains trajectories from non-standard road geometries that are not typically encountered in most driving contexts, making it less suitable for general application.
  - **High Non-vehicular Traffic:** The predominance of pedestrians and cyclists in the uniD dataset introduces a bias towards non-motorized traffic, which is not the focus of this thesis aimed at vehicle dynamics and automated driving systems.

5.1.1.2. Implications for Synthetic Data Generation

The use of inD and rounD datasets provides a robust platform for generating synthetic data through generative models. By encompassing a wide range of real-world driving conditions and interactions, these datasets help in training models that can produce highly realistic, diverse, and temporally coherent traffic scenarios. These scenarios are essential for:

- Validating the performance and safety of autonomous driving systems across various complex urban and roundabout intersections.

- Enhancing the decision-making algorithms of autonomous vehicles by exposing them to a multitude of challenging driving conditions encapsulated in these datasets.

The exclusion of simpler or less relevant datasets ensures that the generative models focus on learning the most critical and frequent scenarios likely to be encountered by vehicles in real-world settings, thereby maximizing the relevance and applicability of the generated synthetic data.

5.1.2. Test Cases Extraction

In the development of synthetic datasets for autonomous vehicle systems, selecting appropriate test case sizes, feature sets, and methods for extracting challenging scenarios is crucial. These decisions directly impact the quality of training data and, consequently, the performance of the generative models used. This section outlines the methodology employed to optimize data processing for generating realistic traffic scenarios using the inD and rounD datasets.

5.1.2.1. Case Size Determination

The decision to use 500 frames per test case, corresponding to approximately 20 seconds of real-world activity, was based on the typical visibility duration of vehicles within the datasets' observed intersections and roundabouts. Most vehicles appear for about 300-400 frames (15-20 seconds), thus a case length of 500 frames ensures complete vehicle trajectories are captured without excessive padding, which can obscure the learning process with irrelevant data.

5.1.2.2. Feature Selection

The positional coordinates (x, y) of each track were selected as the primary features. This decision was informed by the premise that velocity and orientation could be implicitly derived from these coordinates, allowing the model to focus on learning fundamental aspects of vehicle dynamics without additional noise from less consistent data.

5.1.2.3. Sliding Window and Data Diversity

A sliding window of 100 frames was implemented between successive cases to balance data diversity against computational efficiency. This window size sufficiently reduces data overlap and replicated cases, minimizing unnecessary computational overhead while ensuring a diverse set of scenarios is captured for model training.

5.1.2.4. Extraction of Challenging Cases

Challenging cases-those that are most informative for training purposes-are identified using a density filter where the track length exceeds 90% of the case length. This

criterion ensures the inclusion of complete and complex trajectories. In scenarios with more than five tracks, tracks are prioritized and selected based on their length to maintain focus on the most relevant and challenging data.

5.1.2.5. Map-Based Data Segregation

Data was segregated based on the specific map layouts associated with each dataset to tailor the synthetic data generation to realistic, location-specific driving conditions. This approach enhances the geographical relevance of the training data, which is pivotal for the practical application of the models in real-world scenarios.

5.1.3. Processing the Document from IKA Datasets

The IKA dataset, comprising multiple CSV files, provided a robust platform for analyzing vehicular and pedestrian traffic patterns. In order to prepare the dataset for gen AI models, specific preprocessing steps were necessary, particularly concerning the ‘tracksMeta.csv‘ and ‘tracks.csv‘ files.

In the "tracksMeta.csv" file, the initial task was to filter out abnormal data entries. This involved removing tracks that were anomalously long, which were identified by their excessive number of frames. Such tracks could skew the analysis and needed to be excluded to ensure the accuracy of the results. Additionally, I filtered out pedestrian-related tracks and other irrelevant data types. This was achieved by scrutinizing the "numFrames" attribute, which was pivotal in building the "row pointer" for efficient data referencing in the subsequent steps.

tracksMeta.csv

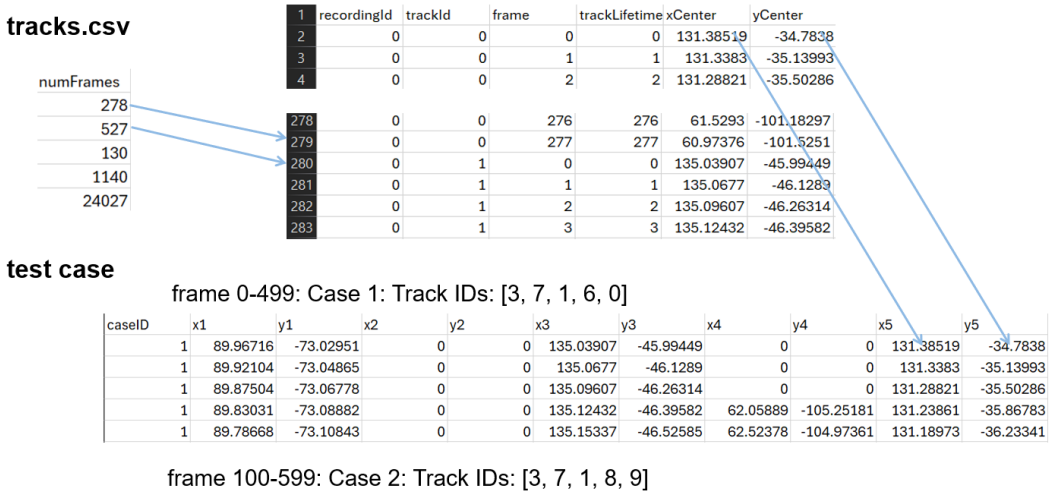
recordingId	trackId	initialFrame	finalFrame	numFrames	width	length	class
0	0	0	277	278	1.76028	4.03141	car
0	1	0	526	527	1.84093	3.92792	car
0	2	0	129	130	1.84183	4.26398	car
0	3	0	1139	1140	3.06624	19.08486	truck_bus
0	4	0	24026	24027	1.82684	4.09254	car
0	81	5721	6140	420	2.09761	5.67524	car
0	82	5774	5975	202	1.7826	3.76603	car
0	83	5781	6491	711	0	0	pedestrian
0	84	5802	6549	748	0	0	pedestrian

row pointer for tracks.csv

Figure 5.3.: tracksMeta csv document from IKA datasets

The primary challenge in processing the "tracks.csv" file was managing its vast size, as it contained tens of thousands of rows. Iterating through such an extensive

file repeatedly was impractical due to time and resource constraints. To address this, I implemented a "row pointer" system. This method allowed for direct querying of intervals associated with specific "trackId" values without the need to traverse the entire file. By employing this approach, I could swiftly extract the relevant x and y coordinates for each track.



	recordingId	trackId	frame	trackLifetime	xCenter	yCenter
2	0	0	0	0	131.38519	-34.7838
3	0	0	1	1	131.3383	-35.13993
4	0	0	2	2	131.28821	-35.50286
278	0	0	276	276	61.5293	-101.18297
279	0	0	277	277	60.97376	-101.5251
280	0	1	0	0	135.03907	-45.99449
281	0	1	1	1	135.0677	-46.1289
282	0	1	2	2	135.09607	-46.26314
283	0	1	3	3	135.12432	-46.39582

caseID	x1	y1	x2	y2	x3	y3	x4	y4	x5	y5
1	89.96716	-73.02951	0	0	135.03907	-45.99449	0	0	131.38519	-34.7838
1	89.92104	-73.04865	0	0	135.0677	-46.1289	0	0	131.3383	-35.13993
1	89.87504	-73.06778	0	0	135.09607	-46.26314	0	0	131.28821	-35.50286
1	89.83031	-73.08882	0	0	135.12432	-46.39582	62.05889	-105.25181	131.23861	-35.86783
1	89.78668	-73.10843	0	0	135.15337	-46.52585	62.52378	-104.97361	131.18973	-36.23341

Figure 5.4.: Extract tracks to test cases

Using the extracted coordinates, I constructed test cases which were integral to the subsequent stages of my analysis. This process not only optimized the data handling but also significantly streamlined the analysis, allowing for more focused and effective exploration of traffic patterns within the dataset.

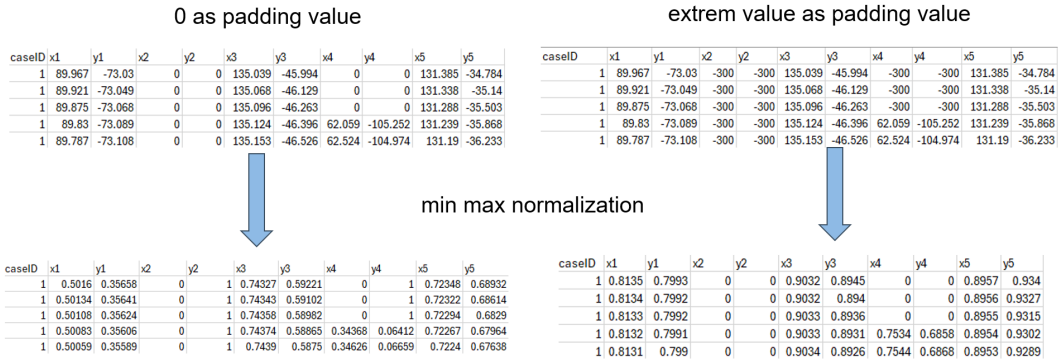
Conclusion on Data Preparation Although preliminary, the data preparation phase is critical as it establishes the foundation for the synthetic data's quality. By carefully selecting case sizes, features, and extraction methods, the training data is optimized to enhance the generative models' learning efficacy and effectiveness, thereby producing realistic and practical synthetic time series data for autonomous driving applications.

5.2. Training

In my research, I focused on applying and optimizing generative models, particularly for the generation of time-series data. By utilizing and enhancing multiple open-source repositories, I was able to explore and improve the performance of these models.

5.2.1. Normalization

During the preprocessing of the IKA dataset, a significant challenge was managing missing data points in some tracks at certain time intervals. Initially, I adopted a straightforward approach of using 0 as the padding value. However, this method presented complications during the normalization phase, particularly because the original dataset contained both positive and negative values. After normalization, zeros could transform into 0 or 1, depending on the applied scaling. This transformation risked introducing confusion in the model, as it might treat these normalized zeros as outliers or significant data points.



caseID	x1	y1	x2	y2	x3	y3	x4	y4	x5	y5
1	89.967	-73.03	0	0	135.039	-45.994	0	0	131.385	-34.784
1	89.921	-73.049	0	0	135.068	-46.129	0	0	131.338	-35.14
1	89.875	-73.068	0	0	135.096	-46.263	0	0	131.288	-35.503
1	89.83	-73.089	0	0	135.124	-46.396	62.059	-105.252	131.239	-35.868
1	89.787	-73.108	0	0	135.153	-46.526	62.524	-104.974	131.19	-36.233

caseID	x1	y1	x2	y2	x3	y3	x4	y4	x5	y5
1	89.967	-73.03	-300	-300	135.039	-45.994	-300	-300	131.385	-34.784
1	89.921	-73.049	-300	-300	135.068	-46.129	-300	-300	131.338	-35.14
1	89.875	-73.068	-300	-300	135.096	-46.263	-300	-300	131.288	-35.503
1	89.83	-73.089	-300	-300	135.124	-46.396	62.059	-105.252	131.239	-35.868
1	89.787	-73.108	-300	-300	135.153	-46.526	62.524	-104.974	131.19	-36.233

caseID	x1	y1	x2	y2	x3	y3	x4	y4	x5	y5
1	0.5016	0.35658	0	1	0.74327	0.59221	0	1	0.72348	0.68932
1	0.50134	0.35641	0	1	0.74343	0.59102	0	1	0.72322	0.68614
1	0.50108	0.35624	0	1	0.74358	0.58982	0	1	0.72294	0.6829
1	0.50083	0.35606	0	1	0.74374	0.58865	0.34368	0.06412	0.72267	0.67964
1	0.50059	0.35589	0	1	0.7439	0.5875	0.34626	0.06659	0.7224	0.67638

caseID	x1	y1	x2	y2	x3	y3	x4	y4	x5	y5
1	0.8135	0.7993	0	0	0.9032	0.8945	0	0	0.8957	0.934
1	0.8134	0.7992	0	0	0.9032	0.894	0	0	0.8956	0.9327
1	0.8133	0.7992	0	0	0.9033	0.8936	0	0	0.8955	0.9315
1	0.8132	0.7991	0	0	0.9033	0.8931	0.7534	0.6858	0.8954	0.9302
1	0.8131	0.799	0	0	0.9034	0.8926	0.7544	0.6868	0.8953	0.9289

Figure 5.5.: Normalization

To address this issue, I experimented with using an extreme value as a padding strategy. Typically, I chose values significantly less than the minimum observed value in the dataset, often two to three times lower. This approach aimed to differentiate padded values distinctly from actual data points during and after normalization, reducing potential confusion within the model.

Additionally, I explored using actual observed values as padding, which intuitively could have offered a more seamless integration of missing points. However, this method resulted in unstable outputs and compounded the challenge of distinguishing between real and padded data in subsequent analyses.

Given these experiences, the choice between using zero and extreme values for padding required careful consideration based on the characteristics of the source data. Each approach had to be tailored to minimize its impact on the overall dataset's distribution and to preserve the integrity of the subsequent analyses. This nuanced approach to handling missing data and normalization was crucial for maintaining the quality and reliability of the traffic pattern analysis in the IKA dataset.

5.2.2. TimeGAN Training Process

For the TimeGAN model, I randomly selected time series data from the dataset for training. The parameters used were as follows:

- **Epochs:** Set between 5000 and 10000 to ensure sufficient depth of training and to observe long-term learning trends.
- **Normalization:** I experimented with both min-max and standard normalization (using mean and variance of the data).
- **Batch Size:** Set at 128, balancing memory capacity and training efficiency.
- **Hidden Dimensions:** Set the embedding layer's hidden dimensions to 20 to capture the intrinsic features of the data.
- **Discriminator Threshold:** Set at 0.15 to judge the authenticity of the generated sequences.
- **Number of Layers:** Three layers, balancing model complexity and computational demands.
- **Optimizer:** Used the Adam optimizer, which generally performs better with non-stable objective functions.
- **Learning Rate:** Set at 0.001 to balance convergence speed and model stability.
- **Sequence Length:** Tested with lengths of 100, 250, and 500 to evaluate the impact on model performance.
- **Data Split:** Divided the dataset into 50% training and 50% testing to effectively validate the model.

The TimeGAN implementation and optimizations were based on the repositories found at `https://github.com/jsyoon0823/TimeGAN` and `https://github.com/birdx0810/timegan-pytorch`. My fork of the repository can be accessed at `https://github.com/RuizhangZhou/timegan-pytorch`.

5.2.3. Diffusion-TS Training Process

For the Diffusion-TS model, I focused on long-duration training to optimize performance, utilizing the following parameters:

- **Epochs:** Set between 20000 and 40000 to fully learn and simulate complex time series data.

- **Padding Value:** Experimented with padding values of 0, actual data minimum negatives, and less than three times the negative of the data minimum to explore the impact of different normalization methods.
- **Layer Configuration:** Encoder and decoder configured with 3 and 2 layers, respectively, to build sufficient model complexity and improve generation quality.
- **Beta Schedule:** 'Cosine' used as the beta scheduling scheme to optimize variance decay during training.
- **Number of Heads:** Set the number of heads in the multi-head attention mechanism to 4 to enhance the model's attention capabilities.
- **Batch Size:** As with TimeGAN, set to 128.
- **D_model:** Embedding dimension set to 96 to provide adequate capacity to handle complex data.
- **Base Learning Rate:** Set at 1.0e-5 considering the stability and convergence over long training durations.
- **Data Split:** Divided the dataset into 80% training and 20% testing, aiding in maintaining model generalization while providing a more precise evaluation of model performance.

The Diffusion-TS model implementation and optimizations were based on the repository found at `https://github.com/Y-debug-sys/Diffusion-TS`. My fork of the repository can be accessed at `https://github.com/RuizhangZhou/Diffusion-TS`.

5.2.4. Hardware and Software Configuration

For the training and implementation of the generative models, I utilized high-performance hardware and software. The hardware included NVIDIA H100 and Quadro RTX 6000 GPUs. These GPUs are well-suited for deep learning tasks due to their high computational power and efficiency in processing large datasets, which significantly accelerates the training process.

- **NVIDIA H100 GPU:** Known for its robust architecture and AI optimization capabilities, this GPU provided the necessary computational power to handle complex model training with extensive data.
- **Quadro RTX 6000 GPU:** Renowned for its large memory capacity and excellent performance in graphics-intensive and machine learning tasks, which enhances the training stability and speed.

On the software side, I employed Python as the programming language with the PyTorch framework. PyTorch is favored for its dynamic computational graph and efficient memory usage, which are critical for developing and training advanced machine learning models. Its intuitive syntax and support for GPU acceleration are key factors that facilitated the rapid prototyping and experimentation of various model architectures and optimizations.

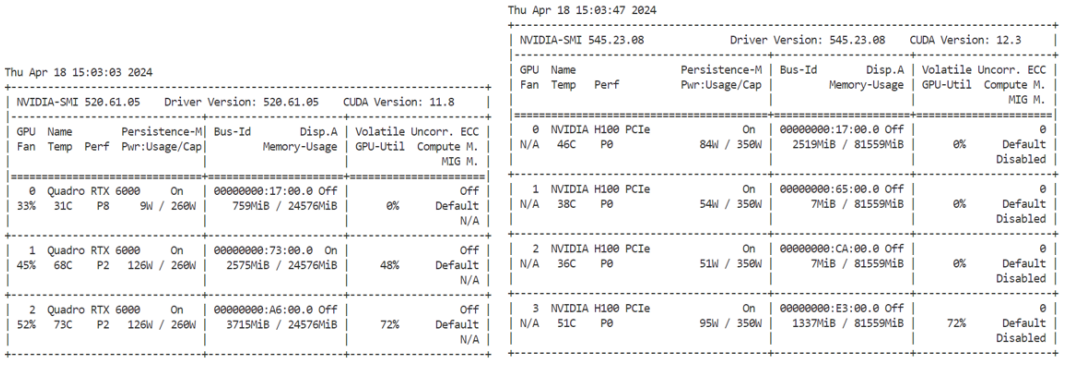


Figure 5.6.: Training hardwares

5.2.5. Training Process

The visualization and analysis of training losses were managed using TensorBoard, which enabled clear tracking and visualization of loss metrics.

5.2.5.1. TimeGAN Loss

For TimeGAN, the losses visualized included Embedding Loss and Supervised Loss, both of which typically converged within tens of minutes, indicating stable training conditions. Conversely, the Unsupervised Loss, related to the Discriminator and Generator in the GAN setup, exhibited significant fluctuations. This behavior is common in GAN training due to the adversarial nature of the process, where the Discriminator and Generator often demonstrate an inverse correlation and can show cyclical variations. Given the lack of a single loss metric that indicates the optimal GAN model, selecting the best model required ongoing evaluation based on the generation quality, loss balance, and training stability. This often meant several hours of training with periodic reviews of checkpoints and visual outputs to decide on the termination of the training process.

5.2.5.2. Diffusion-TS Loss

In contrast, the Diffusion-TS model features a simpler loss configuration. I plotted the loss changes myself using the recorded Mean Square Error (MSE) data. The simplicity of using MSE facilitated a clear determination of model convergence, typically achieved within about five hours. This faster convergence compared to TimeGAN can be attributed to the direct feedback mechanism of MSE, which straightforwardly assesses the discrepancies between generated and actual data.

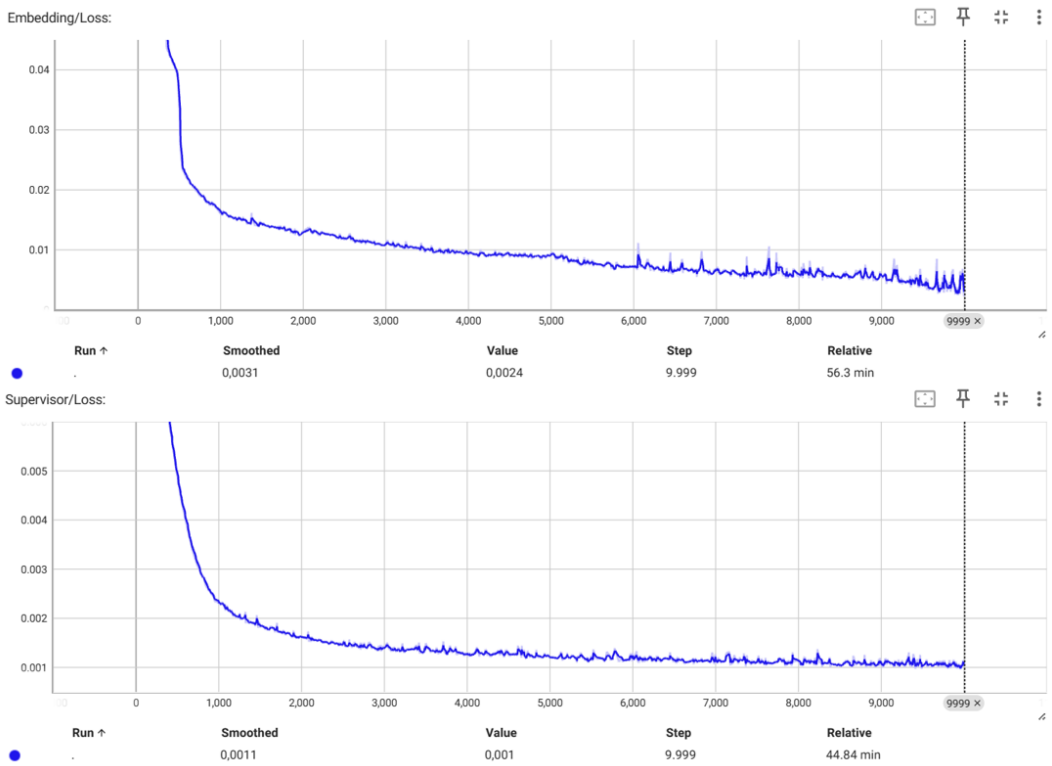


Figure 5.7.: timeGAN: embedding loss and supervised loss

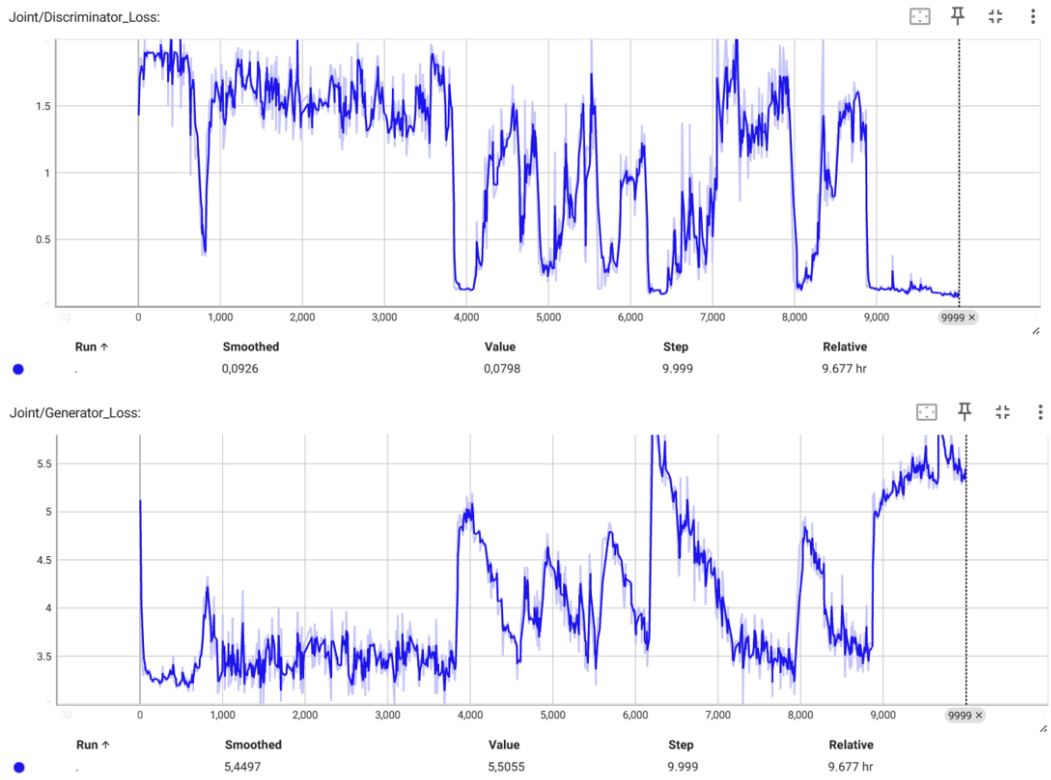


Figure 5.8.: timeGAN: unsupervised loss

Figure 5.9.: Diffusion Time-Series loss

5.3. Integration into CPM Remote

5.3.1. Convert the data format

The Data Converter plays a pivotal role in transforming real-world trajectory data into usable simulation scenarios for the CPM Lab. This conversion is essential for utilizing the rich trajectory datasets such as inD and rounD, which encompass diverse traffic scenarios from German roads, particularly intersections and roundabouts, respectively.

Figure 5.10.: Data converter

5.3.1.1. Importing Data Sets

The initial step involves importing trajectory data from the inD and rounD datasets. These datasets provide comprehensive records of vehicle, pedestrian, and cyclist movements, captured through drone-based aerial photography. The data, originally stored in CSV format, includes detailed positional coordinates along with time stamps, offering a granular view of various traffic dynamics.

5.3.1.2. Replay of Recordings

To facilitate the accurate replay and visualization of these recordings, the DSC enables the scenarios to be played back within the simulation environment. This replay capability is crucial for verifying the data's integrity and for preliminary assessments before the actual conversion process.

5.3.1.3. Defining Spatial Transformations

The converter applies several spatial transformations to align the data with the simulation's coordinate system:

- **Translation** adjusts the origin of the coordinate system to match that of the simulation environment.
- **Rotation** corrects the orientation of the data to comply with the directional flows used within the simulations.

- **Scaling** modifies the scale of the data to fit the simulation model dimensions, which is critical especially when transitioning from real-world measurements to a scaled model environment.

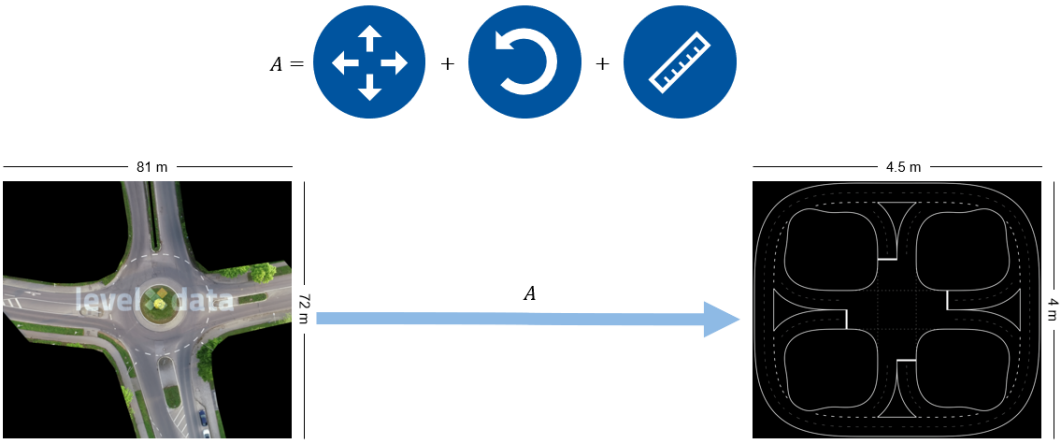


Figure 5.11.: Data converter: spatial mapping

5.3.1.4. Exporting Scenarios

Post transformation, the processed data is exported into XML format, which details the scenario configurations and the dynamic and static elements involved, such as vehicles and road layouts. These XML files are then used to create interactive and reproducible traffic scenarios in the CPM Lab, providing a platform for extensive testing and development of traffic management solutions.

These capabilities of the Data Set Converter ensure that the scenarios generated are not only accurate reflections of real-world conditions but are also optimized for the high-fidelity simulations required in modern traffic research and autonomous vehicle development.

Figure 5.12.: Exporting Scenarios

5.3.2. Integration into CPM Remote

The integration of XML formatted scenarios into the CPM Remote project involves several critical steps that ensure the scenarios are not only accurately represented but also fully functional within the CPM Lab environment. This process enhances the flexibility and applicability of the CPM Remote project, allowing for extensive testing and development of autonomous vehicle maneuvers in a simulated environment.

5.3.2.1. Scenario Format and Conversion

Scenarios for CPM Remote are provided in an XML format, which structurally defines the various elements of traffic simulations, such as vehicle paths, interactions, and environmental conditions. The process of integrating these scenarios into CPM Remote involves:

- **XML Scenario Definition:** Each scenario is defined in an XML file, which specifies the essential components of the simulation, including vehicle types, initial positions, paths, and behaviors over time. These XML files are generated using the Data Set Converter tool, which extracts and converts real-world traffic data (from datasets like inD and rounD) into structured XML scenarios.

- **Spatial Transformation:** The scenarios undergo spatial transformations to fit the specific configurations of the CPM Lab. This includes scaling, rotation, and translation adjustments to align the scenarios with the lab's coordinate system and scale. These transformations are crucial for maintaining the fidelity and relevance of the simulation data to real-world conditions.

- **Data Set Converter Utilization:** The Data Set Converter (DSC), a software tool developed for CPM Remote, plays a pivotal role in this process. It not only performs the initial data extraction and conversion but also allows for the fine-tuning of the scenarios through user-defined parameters such as cropping times, scaling factors, and rotation angles.

5.3.2.2. Integration and Simulation Workflow

Once the XML scenarios are prepared, they are integrated into the CPM Remote's simulation environment through the following steps:

- **Importing Scenarios:** The XML files are imported into CPM Remote, where they are parsed and stored in a structured format within the project's database. This includes mapping all scenario elements to the corresponding simulation entities and parameters.

- **Simulation Execution:** In CPM Remote, users can load and execute these scenarios to test various vehicle behaviors and traffic management strategies. The simulation leverages the XML-defined paths and interactions to render dynamic, visually accurate representations of traffic conditions as specified by the scenarios.

- **Feedback and Iteration:** Based on simulation outcomes, users can modify scenario parameters or vehicle strategies. The flexibility of the XML format allows for quick iterations and modifications, which are crucial for the experimental development cycles typical in autonomous vehicle research.

5.3.2.3. Export and Reuse

The structured nature of XML also facilitates the export and reuse of scenarios across different projects or simulations within CPM Remote:

- **Scenario Reusability:** Once scenarios are converted into XML and integrated into CPM Remote, they can be easily reused or shared across different simulation projects, enhancing collaborative development and testing.

- **Standardization Benefits:** Using XML allows for standardization in scenario formats, which simplifies the integration of external data sets and improves compatibility with various simulation tools and frameworks used within the CPM Remote ecosystem.

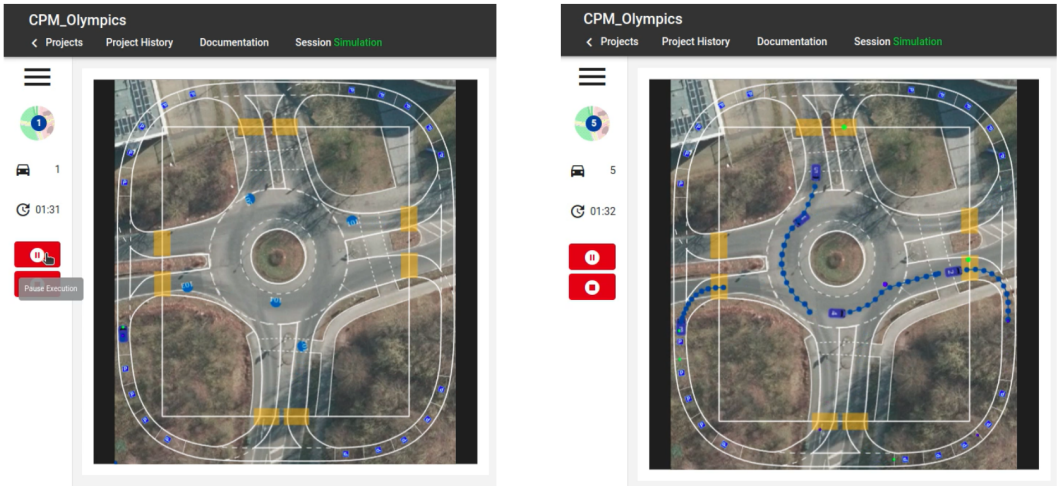


Figure 5.13.: CPM Olympic

These integration processes underscore the importance of XML formatted scenarios in bridging real-world data with simulated environments, thus providing a robust framework for testing and developing autonomous vehicle technologies within CPM Remote.

6. Evaluation

6.1. Qualitative Analysis

The qualitative analysis primarily focuses on the visual inspection of generated traffic scenarios using the trained models on the IKA datasets. This section provides insights into the characteristics and quality of the synthetic data, drawing comparisons across different models and datasets.

6.1.1. Visualization of Generated Cases

Generated datasets are tailored to match the diversity and scale of the training data. This ensures that the complexity and variability in the training scenarios are adequately captured and reflected in the synthetic outputs. For instance, visualizations from the rounD00 map, generated after training on this specific dataset, reveal patterns and interactions that are typical in roundabouts.

6.1.2. Observations from Different Models

Initial trials with TimeGAN on the inD dataset exhibited significant tremble and instability in the vehicle trajectories, making it challenging to produce decent results. The trajectories were often jittery, which could lead to unrealistic behavior patterns unsuitable for robust simulation purposes. This prompted a pivot towards Diffusion-TS, a model that demonstrated a higher capacity for generating stable and coherent trajectories.

6.1.3. Comparative Analysis

Upon shifting the focus to Diffusion-TS, extensive tests were conducted across multiple datasets, including inD, rounD, and exiD. The model was tasked with generating 20-second long scenarios (500 time frames), which allowed for the observation of complete vehicle trajectories from entry to exit within the scene. The visualization of these cases revealed that:

- **inD Dataset:** The generated trajectories on the inD dataset were smoother compared to those from TimeGAN, yet minor fluctuations were still observable, indicating slight trembles.

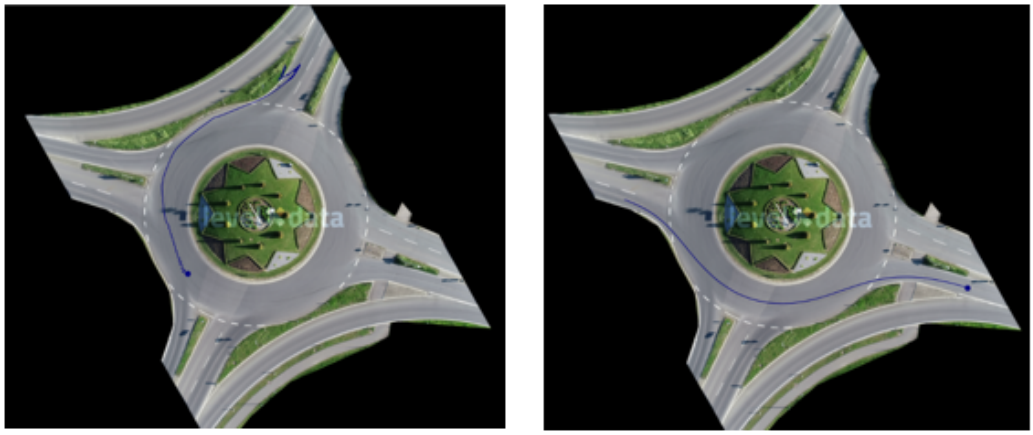


Figure 6.1.: TimeGAN vs Diffusion-TS on 1 vehicle case

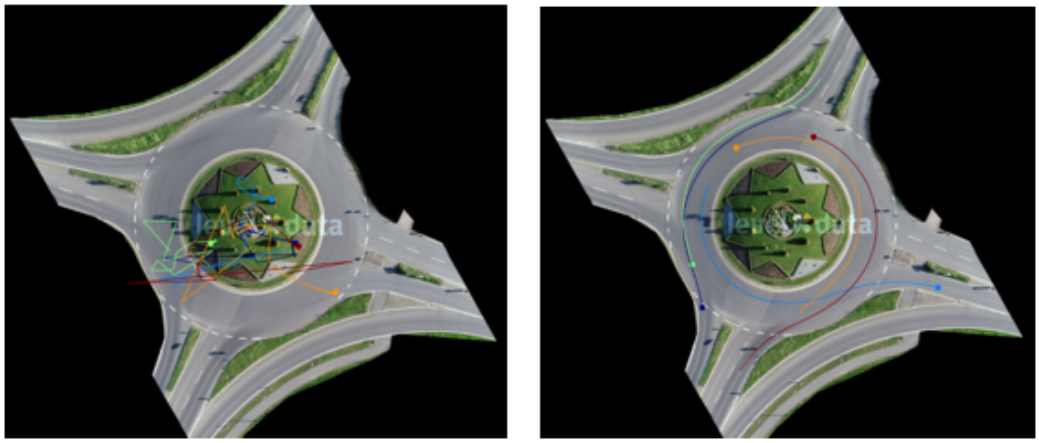


Figure 6.2.: TimeGAN vs Diffusion-TS on 5 vehicles case

- **rounD Dataset:** The scenarios generated on the rounD dataset closely mimicked realistic roundabout maneuvers, showcasing the model's ability to replicate complex driving behaviors in circular trajectories.

- **exiD Dataset:** Testing on the exiD dataset highlighted the model's capability to handle high-speed merging and diverging movements, although the scenarios were less challenging than those involving intricate interactions in urban settings.

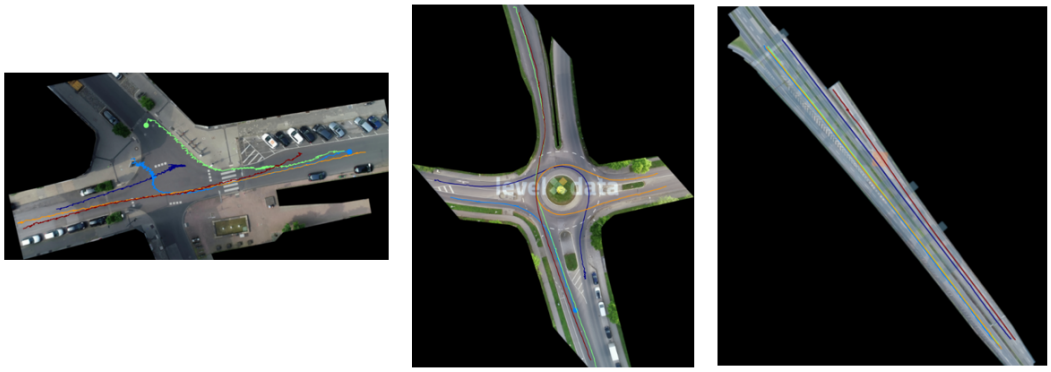


Figure 6.3.: Diffussion-TS on inD, rounD and exiD datasets

6.1.4. Implications for Autonomous Vehicle Testing

The qualitative assessments suggest that Diffusion-TS can significantly contribute to the generation of realistic, dynamic, and varied driving scenarios, essential for training and evaluating autonomous driving systems. By providing a rich dataset of synthesized but plausible traffic situations, these models help enhance the robustness of motion planning algorithms under diverse and challenging conditions.

6.2. Quantitative Analysis

This section delves into the quantitative evaluation of the synthetic data generated by the model, utilizing several advanced statistical and visualization techniques to ensure the diversity and realism of the synthetic data relative to the real-world data on which the model was trained.

6.2.1. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a linear dimensionality reduction technique used to reduce the multi-dimensional data sets to lower dimensions for analysis, while retaining the essential data characteristics that contribute most to its variance. In the context of this thesis:

- PCA reduces the high-dimensional feature space of the vehicle trajectories (which includes features like position, speed, and acceleration) to 2 principal components.

- This reduction simplifies the visualization while preserving the structure and relationships inherent in the data, thus facilitating a clearer understanding of the underlying patterns.

6.2.2. t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-Distributed Stochastic Neighbor Embedding (t-SNE) is a non-linear dimensionality reduction technique particularly well-suited for the visualization of high-dimensional datasets. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. The process involves:

- Constructing a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a high probability of being picked while dissimilar objects have an extremely low probability of being picked.
- Defining a similar probability distribution over the points in the low-dimensional map, which helps in preserving the local structure of the data.
- The KL divergence between these two distributions is minimized during training via gradient descent.

6.2.3. Density Analysis

Density analysis involves examining the distribution of the generated data to ensure it covers the entire spectrum of the training data's distribution. For this analysis:

- All coordinate values, including outliers, are normalized using min-max scaling to the [0,1] range to standardize the data before processing.
- This normalization helps in mitigating the influence of extreme values which can skew the analysis and affect the model's performance.

6.2.4. Evaluation Metrics

To quantitatively assess the quality of the generated datasets, this study employs two main metrics:

- **Inception Score (IS):** This score measures both the diversity and the quality of the generated images. A higher IS indicates better model performance.
- **Frechet Inception Distance (FID):** This metric calculates the distance between feature vectors calculated for real and generated images. Lower FID values suggest more realistic and varied output.

6.2.5. Model Comparisons

Utilizing these analytical techniques, the study evaluates the performance of diffusion-based generative models against TimeGAN. Preliminary results indicate that while TimeGAN struggles with maintaining stable generation over longer sequences (e.g., 10 seconds of vehicle data), diffusion models show promising capabilities in handling longer sequences with better stability and diversity, particularly evident in datasets such as inD and rounD.

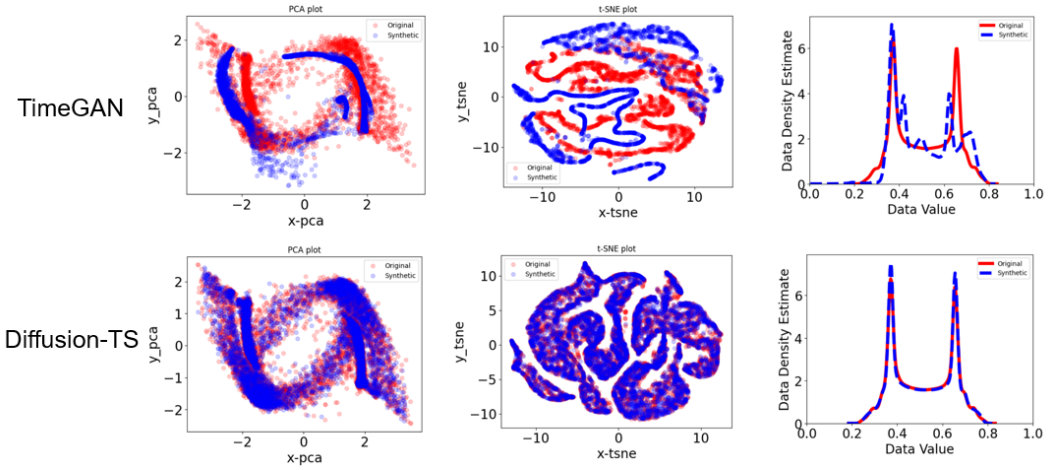


Figure 6.4.: timeGAN vs Diffusion-TS: distribution between original and sythetic datas

6.3. Results

The quantitative analysis underscores the efficacy of diffusion models in synthesizing realistic, diverse traffic scenarios that can potentially accelerate the development and testing of autonomous driving systems, thereby contributing substantially to advancements in vehicle automation technology.

This part has explored the capabilities of generative models, specifically focusing on diffusion models, to synthesize realistic time-series data for traffic scenarios, which is critical for advancing autonomous driving technologies. The following points summarize the key findings and implications of this research:

6.3.1. Complexity in Learning Different Track Types

The ability of generative models to learn and replicate complex behaviors varies significantly across different track types:

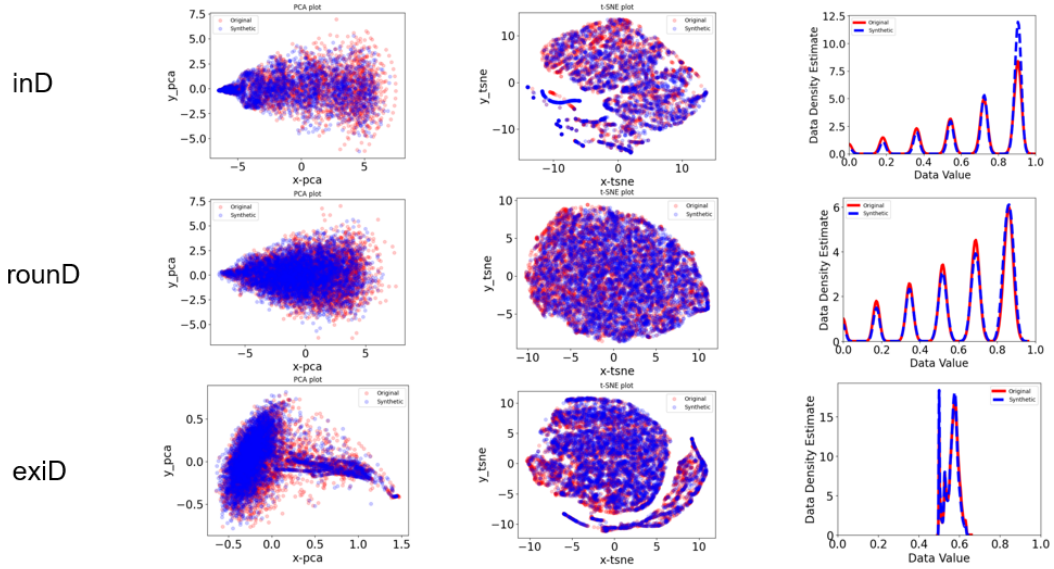


Figure 6.5.: Diffusion-TS: distribution between original and sythetic datas on inD, rounD and exiD datasets

- **Pedestrians:** Tracks involving pedestrians, characterized by their irregular paths and unpredictable motion patterns, present considerable challenges in modeling. These difficulties are exacerbated when pedestrian data is mixed with vehicular tracks, which follow more predictable and regular trajectories.
- **Vehicles:** In contrast, vehicles, which typically exhibit more structured movements (such as consistent speeds and directions), are easier to model. The predictability of vehicle trajectories aids in training more stable and reliable generative models.

6.3.2. Impact of Increasing Vehicles

An increase in the number of vehicles within the training data introduces complexity primarily due to the enlarged feature space:

- The addition of each vehicle (or track) proportionally increases the dimensions of the feature space, which complicates the learning process by requiring the model to capture interactions across a broader array of elements.
- This dimensional increase necessitates more sophisticated model architectures and training regimens to effectively learn the underlying patterns without succumbing to overfitting or computational inefficiencies.

6.3.3. Comparison Between Generative Models

The research conducted has provided comparative insights into the performance of different generative models:

- **Diffusion Models:** These models have demonstrated a superior ability to train effectively, achieve convergence reliably, and produce high-quality outputs that closely mimic the real-world data. The ease of training and the quality of the generated data suggest that diffusion models are particularly well-suited for generating complex multi-track scenarios.
- **TimeGAN:** While TimeGAN is a potent model for time-series generation, it struggles with stability and consistency, especially as the sequence length and complexity increase.

6.3.4. Performance Across Datasets

The performance of the models varies across different datasets, which can be ranked based on the complexity of vehicular dynamics they contain:

- The **exiD dataset**, which contains highly interactive and complex driving scenarios, was found to be the most conducive for training effective models.
- The **rounD** and **inD datasets** presented moderate challenges, with rounD providing a slightly more complex environment than inD due to its roundabout scenarios.
- Datasets like **highD** and **uniD** were considered either too simplistic or overly complex for use in standard CPM Olympic test cases, indicating a need for balanced complexity in training datasets.

Overall, this part underscores the potential of diffusion models as robust tools for generating synthetic datasets that are not only realistic and diverse but also adaptable to various complexities within traffic scenarios. These models hold promise for significantly enhancing the data availability for training and testing autonomous driving systems, thus accelerating the development of safe and efficient autonomous vehicles.

7. Conclusion

The aim of this document was to provide students at the end of their studies with a template for their written thesis. Due to the common lack of experience in the field of academic writing this work is intended to provide a template for structured writing. Therefore, this entire document is structured as a thesis should usually be.

Moreover it provides some hints how citations should be used and how figures should be dealt with to achieve high quality versions of latex documents. In contrary to this sentence, a conclusion should not introduce new information, about the topics discussed before, which has not yet been presented.

The following (optional) section provides some further ideas for potential extension of this work.

7.1. Future Work

The explorations and findings from this thesis pave the way for several promising directions for future research, which could further enhance the capabilities and applicability of generative models for synthetic traffic data generation. The following are some of the areas where future work could be particularly fruitful:

7.1.1. Independent Track Learning

Future models could benefit from an approach that involves:

- **Separate Learning of Vehicle Tracks:** Initially, learn the trajectories of individual vehicles independently to capture the unique behavioral patterns of different vehicle types more effectively.
- **Combination Based on Distribution Characteristics:** After learning individual tracks, these can be combined based on their distributional characteristics to simulate realistic multi-vehicle interactions.

7.1.2. Multivehicle Information Processing

Understanding and processing multivehicle dynamics from single-map inputs offer a compact yet rich source of behavioral data:

- Positional information from multiple vehicles on a single map can implicitly include dynamic data such as speed and directional changes, which are crucial for modeling realistic vehicular interactions.

7.1.3. Single Vehicle Information from Different Maps

- **Extraction and Finetuning:** Extract single vehicle's speed and direction information across various maps to build a generalized understanding of vehicle dynamics, which can then be fine-tuned to specific maps or scenarios.

7.1.4. Improvement in Outlier Handling

- **Enhanced Outlier Management:** Improve the methods for managing outliers in the data, which include zero or extreme values that can skew the model performance. Refining how these outliers are handled will lead to more robust and reliable model outputs.

7.1.5. Implementation Strategy

- Given the current focus on modeling interactions among multiple vehicles, a stratified approach where each map is used to train distinct models could be explored. Similar to the methodologies employed in training large language models, this would involve:
    1. **Primary Model Training:** Develop a comprehensive model on aggregated data to learn general behaviors and interactions.
    2. **Specialized Fine-Tuning:** Subsequently, fine-tune this model on specific maps or for particular scenarios to tailor the behaviors to fit more localized or specialized contexts.

In summary, the future work will aim to separate and then integratively learn the diverse aspects of vehicular motion and interactions, enhancing the granularity and accuracy of the generated synthetic data. Such advancements could significantly improve the developmental frameworks for autonomous driving systems, leading to safer and more efficient vehicular technologies.

Bibliography

A. Appendix1