Deep Learning Software for Traffic State Prediction

Center of Excellence for Mobility and Congestion - Theme 1
NCDOT Research Project TCE2020-02

August 2023

Authors

Dr. Sambit Bhattacharya
Dr. Hyoshin Park
Mr. Daniel Rundell

NC STATE UNIVERSITY
ITRE
Institute for Transportation Research and Education

RESEARCH & DEVELOPMENT

Technical Report Documentation Page

1. Report No. FHWA/NC/TCE2020-02 (P1)	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Deep Learning Software for Traffic State Prediction		5. Report Date August 2023
		6. Performing Organization Code
7. Author(s) Dr. Sambit Bhattacharya, Dr. Hyoshin Park, Mr. Daniel Rundell		8. Performing Organization Report No.
9. Performing Organization Name and Address North Carolina State University Institute for Transportation Research and Education Campus Box 8601, Raleigh, NC 27695-8601		10. Work Unit No. (TRAIS) 11. Contract or Grant No.
12. Sponsoring Agency Name and Address North Carolina Department of Transportation Research and Development 1549 Mail Service Center Raleigh, NC 27699-1549		13. Type of Report and Period Covered 2/1/2020 – 2/28/2023 14. Sponsoring Agency Code TCE2020-02 (P1)
Supplementary Notes:		

16. Abstract

As urban populations continue to grow, traffic congestion has become an increasingly pressing issue for drivers, leading to a greater demand for advanced navigation aids and traffic management solutions. Real-time traffic information, advanced traffic signal control strategies, and the integration of various data sources can play a crucial role in improving traffic flow and safety. However, these advancements also present challenges, such as data integration and effective analysis. This research explored the development of deep neural network-based software that converts video into structured data formats, providing valuable information on driving conditions and traffic incidents. The use of synthetic data in response to the absence of normal traffic patterns during the COVID-19 pandemic is also examined. The research products of this project include a video analytics pipeline, robust data fusion techniques, and a prototype application designed to work with actual traffic control hardware, ultimately aiming to enhance traffic flow and safety on our roads.

17. Key Words Traffic congestion, traffic signal control, sim-to-real, image processing, data fusion	18. Distribution Statement		
19. Security Classif. (of this report) Unclassified	20. Security Classif. (of this page) Unclassified	21. No. of Pages 81	22. Price

Form DOT F 1700.7 (8-72) **Reproduction of completed page authorized**

Disclaimer

The contents of this report reflect the views of the author who is responsible for the facts and the accuracy of the data presented herein. The contents of the report do not reflect the official views or policies of the North Carolina Department of Transportation or the Federal Highway Administration. This report does not constitute a standard, specification, or regulation.

Acknowledgments

We would like to express our sincere gratitude to the North Carolina Department of Transportation (NCDOT) for funding this research project. Without their support, this study would not have been possible.

We would also like to extend our heartfelt thanks to all the faculty members, students, and staff who contributed to this project. Their tireless efforts and dedication were instrumental in ensuring the success of this project.

The project senior personnel:

Sambit Bhattacharya – Fayetteville State University – Principal Investigator
Murat Adivar – Fayetteville State University – Co-Principal Investigator
Ali Hajbabaie – NC State University – Co-Principal Investigator
Noel Greis – NC State University – Co-Principal Investigator
George List – NC State University – Researcher
Thomas Chase – NC State University – Researcher
Hyoshin (John) Park – NC A&T State University – Co-Principal Investigator

The students and research assistants:

Daniel Rundell – Fayetteville State University
Catherine Spooner – Fayetteville State University
Joshua Adams – Fayetteville State University
Ramin Niroumand - NC State University
Niharika Deshpande - NC A&T State University

We also wish to acknowledge the support and guidance provided by the various departments and administrative staff, who helped us to navigate the logistics of this project and to overcome the challenges that we encountered along the way. Thanks to NC State University's Institute for Transportation Research and Education (ITRE) and the ITRE Director Dr. Billy Williams who provided management and research advice.

Finally, we would like to thank our families and loved ones for their unwavering support and understanding throughout the project. Their encouragement and patience were essential in allowing us to pursue our research goals while balancing our personal and professional responsibilities.

Once again, we extend our sincere thanks to everyone who contributed to this project, and we hope that the findings and recommendations from this research will be useful in improving transportation in North Carolina and beyond.

Summary

Drivers are finding themselves in higher traffic congestion as the number of active drivers increases exponentially with growing populations in urban areas. Navigation aids are becoming more prevalent and essential to aid in avoiding these types of travel delays. Traffic and navigation-based applications have created a way for users to have real time information on potential delays and alternate routes. Even with a passive mode enabled, to ensure higher safety for drivers, average speed can be shared which gives indications of route congestion and delays. In an active mode in-depth detail about conditions and traffic incidents become available to other users but this has an associated risk of driver distractions.

Advanced Traffic Signal Control strategies are being adopted by public agencies responsible for maintaining signalized arterials to address the growing volume of traffic on these roads. At the same time, these agencies are collecting more and more data from a variety of sources, which requires careful analysis to be effectively used. As a result, most traffic signal systems and agency arterial programs tend to focus on one type of data at a time. For example, traffic controllers might use point sensors or other detection methods to actuate phases, while many emerging data sources provide more detailed information about the overall traffic state.

One potential benefit of incorporating more accurate traffic state estimates into advanced traffic controllers is increased accuracy in optimization. By considering a wider range of data, traffic controllers can make more informed decisions about how to adjust signal timing and other factors in order to improve traffic flow. This could lead to a variety of benefits, such as reduced congestion, shorter travel times, and improved safety on the roads.

However, incorporating more data into traffic signal systems also presents some challenges. For one, it can be difficult to integrate data from a variety of sources, particularly if these sources use different formats or have different levels of granularity. Additionally, the sheer volume of data being collected can make it difficult for agencies to effectively analyze and use this information. As a result, it is important for agencies to have the necessary tools and resources to effectively manage and make use of the data they are collecting.

Overall, the adoption of Advanced Traffic Signal Control strategies and the increasing use of data in traffic management are important developments that have the potential to improve traffic flow and safety on our roads. However, these advances also bring with them a number of challenges, which must be carefully managed in order to realize their full benefits.

Collection of real-world data can present unique challenges. In the case of the North Carolina Department of Transportation this has been a particularly challenging task in the recent past. The COVID-19 pandemic and resulting quarantine orders reduced traffic state volumes dramatically. The absence of normal traffic patterns became a driving element in pursuing alternative approaches, in this case synthetic data. Synthesized data are those that are created in a digital environment with sufficient resemblance to real world data that a deep learning model trained with synthetic data performs well when presented with real data. Primary advantages to synthesizing data include machine learning model training in the absence of real data, flexible experimentation with different scenario-based traffic states, and streamlined production-ready model development processes.

The purpose of this project was to develop software based on deep neural networks (DNNs), also known as deep learning, to automatically convert video into structured and more useful data formats. This software will provide information on driving conditions and traffic incidents and will be able to fuse this

data with inferences from loop detectors and Bluetooth sensors. There are many potential benefits to this approach.

One of the main benefits of this software is its ability to reduce the amount of data that needs to be processed and stored. Lengthy segments of video often contain a lot of information that is not particularly useful, so by converting the video into a more structured format, it is possible to significantly reduce the size of the data while still retaining the most relevant information. This data, when fused with other sensor data, can be used by advanced traffic controllers, travelers, or to measure arterial performance.

Another benefit of this software is its ability to automatically mine big video and other sensory data saved over days and weeks to detect traffic incidents. This can support decision-making and reduce the manual efforts of operators by automatically extracting useful information from large amounts of data.

The research products of this project include a video analytics pipeline and robust data fusion techniques at the edge, as well as a prototype application that integrates software and hardware into a simulated advanced traffic signal controller that is designed to work with actual traffic control hardware.

Table of Contents

Table of Figures

Introduction

As urban populations continue to grow, traffic congestion has become an increasingly pressing issue for drivers, leading to a greater demand for advanced navigation aids and traffic management solutions. Real-time traffic information, advanced traffic signal control strategies, and the integration of various data sources can play a crucial role in improving traffic flow and safety. However, these advancements also present challenges, such as data integration and effective analysis. This paper explores the development of deep neural network-based software that converts video into structured data formats, providing valuable information on driving conditions and traffic incidents. The use of synthetic data in response to the absence of normal traffic patterns during the COVID-19 pandemic is also examined. The research products of this project include a video analytics pipeline, robust data fusion techniques, and a prototype application designed to work with actual traffic control hardware, ultimately aiming to enhance traffic flow and safety on our roads.

Background

Traffic flow theory is the study of how vehicles navigate highways and roads. The aim is to model the factors that change the flow of traffic (i.e., number of vehicles that are present, their spacing, and speed) and the physical aspects of the road and the environment around the road and vehicles. Mathematical models, simulation tools, and data analysis techniques are leveraged to model and understand traffic flow. This allows traffic pattern analysis with the ability to examine how modifications to a road network or traffic conditions can affect traffic flow state [13]. Traffic flow theory is viewed through two main lenses: microscopic and macroscopic.

At the microscopic level the focus is on individual vehicles and driver behavior. Key concepts in microscopic traffic flow are:

- **Vehicle Dynamics:** Physical properties and behavior of single vehicles are observed. Properties include acceleration/deceleration and the way that the vehicle turns.

- **Driver Behavior:** Examines the way drivers choose their speed and lane positions and how these behaviors change in various road conditions, the driver's emotional state, and driver perception of the environment.

- **Interactions Between Vehicles:** A vehicle's influence of surrounding vehicles through position, driver gestures, and signaling are referred to as interactions between vehicles.

- **Traffic Control Devices:** Traffic lights, road signs, and markings on the pavement are a few examples of these devices.

- **Traffic Flow Models:** The previously mentioned mathematical models that attempt to illustrate and predict the way vehicles will move on a highway or road. These models allow analysis of traffic patterns and the evaluation of variations in traffic control strategies.

Macroscopic traffic flow zooms out from the individual vehicle(s) to observe the big picture of many vehicles and the way they are moving overall. The aim here is to model and understand the factors that change the flow of traffic. These factors include the total number of vehicles in a particular road network,

speed and spacing, and the physical characteristics of the environment and road. The core concepts of macroscopic traffic flow theory are:

- **Traffic Flow Characteristics:** Traffic flow characteristics look at the overall movement of vehicles in each road, highway, or network. Flow rate, or the number of vehicles passing a given point over a specified interval, and the density, or number of vehicles that are in each length of road at a point in time, are common tools for observation and measurement.

- **Traffic Flow Models:** Traffic flow models are like microscopic traffic flow models just on a much larger scale.

- **Capacity:** Maximum number of vehicles that a road can handle without generating a delay or creating congestion.

- **Congestion:** High enough density of traffic that it causes speed reduction and longer travel times.

- **Traffic Control Strategies:** The plans and responses used to mitigate adverse traffic conditions. These can be thoughtful placement of traffic signals, ramp metering, and lane control.

State of the Art, Science, and Practice

Macroscopic traffic states are defined through three primary measures: flow, density, and speed. These measures are closely related, with flow being equal to the product of speed and density. Constraints that impact one or more of these measures can induce changes in the traffic state, which can be propagated under shockwave theory [2]. However, it can be particularly challenging to model or track interrupted flow conditions, particularly due to additional constraints and the severe delays that drivers often experience on arterial roads.

According to the Federal Highway Administration, there are over 330,000 traffic signals in the United States. Of these, over 75% could be improved by updating their equipment or timing plans. Poor traffic signal timing is a significant contributor to delays on major roadways, accounting for nearly 300 million vehicle-hours of delay. In practice, the measurement of performance in arterials has been led by Automated Traffic Signal Performance Measures (ATSPMs), which use high-resolution loop detector and traffic signal controller data for computation [12].

In recent years, there has been a significant shift in the development of signal control algorithms, particularly in the context of connected and autonomous vehicles (CAVs). New signal control methods have been developed that model cooperative control, as well as control in a partial information environment. These approaches have the potential to improve traffic flow and reduce delays on arterial roads and will likely play a key role in the future of transportation.

Vehicle detection is a crucial step in the vision-based traffic monitoring process using a static camera. Various techniques, such as frame differencing, background subtraction, optical flow, and GMM, have been employed for vehicle detection on highways. Following vehicle detection, tracking moving objects over time in an image sequence is typically performed using features such as points, lines, or blobs.

Common tracking algorithms include Kalman filter, adaptive Kalman filter, and particle filter [16]. The final step in video processing is vehicle classification, which can be performed using deterministic methods, stochastic methods, artificial neural networks, and Support Vector Machines. However, factors such as perspective effects, shadows, camera vibration, and lighting changes can lead to occlusions, which can greatly affect system performance. Therefore, occlusion handling is an important step after vehicle detection. Techniques for reducing occlusions include line-based algorithms, fusion of image frames from multiple cameras, algorithms based on car windshield appearance, and feature-based tracking.

In this context, vehicle detection can refer to various sensors and applications, such as driver assistance systems utilizing a moving camera, or traffic surveillance systems using a static camera. The use of a static camera in traffic surveillance allows for the detection of vehicles as the first step in the vision-based traffic monitoring process [9].

Several techniques have been proposed and implemented for vehicle detection, including frame differencing, background subtraction, optical flow, and Gaussian Mixture Models (GMM). These techniques have been successfully applied in highway scenarios, with some variations in performance depending on the specific conditions and parameters used.

After detecting vehicles, the next step is to track them over time in an image sequence. This is typically achieved by matching objects in consecutive frames using features such as points, lines, or blobs. From these track sequences, different object behaviors can be inferred. In literature, some authors have proposed real-time vision-based traffic flow monitoring systems, which use flow models to count vehicles traveling on each lane and to produce traffic statistics. The most widely used tracking algorithms are Kalman filter, adaptive Kalman filter, and particle filter [16].

Finally, vehicle classification is performed to identify the type of vehicle, such as a car or a truck. Automatic classification methods can be divided into deterministic methods, stochastic methods, artificial neural networks [13], and Support Vector Machine (SVM). These methods have been used in various applications, and their performance can vary depending on the specific conditions and parameters used.

It is worth noting that occlusion handling is an important step in the process, as multiple vehicles can be detected as a single vehicle due to perspective effects, shadows, camera vibration, lighting changes, and other factors. To tackle this problem, several methods have been proposed in the literature, such as line-based algorithms, fusion of image frames acquired from multiple cameras [6], and algorithms based on car windshield appearance, vehicle corner as feature, and feature-based tracking. These methods can improve the performance of the system, allowing for more accurate vehicle detection and classification.

The problem of vehicle re-identification (V-reID) is the identification of a particular vehicle as the same object as one observed on a previous occasion, using multiple cameras with non-overlapping views. This problem has emerged due to the increasing demand for public safety and the widespread use of large camera networks in road networks, university campuses, and streets[19]. Traditional loop detectors and other sensors are expensive and impractical for V-reID in such diverse environments, and it is also a laborious task for security personnel to manually identify a vehicle of interest and track it across multiple cameras.

Computer vision can automate the task of V-reID, which can be broken down into two major modules: vehicle detection and vehicle tracking through multiple cameras. Vehicle detection is a well-established task, but V-reID presents a more challenging problem, as it requires correct matching of multiple images of the same vehicle under intense variations in appearance, illumination, pose, and viewpoint. V-reID is considered a multi-camera tracking problem [24].

Despite the advances in vehicle detection and tracking, V-reID still presents several challenges. These include variations in viewpoint and pose, changes in lighting and weather conditions, and partial occlusions. In addition, the large number of vehicles present in the scene can make it difficult to accurately match a target vehicle across multiple cameras.

Another challenge is the lack of large-scale benchmark datasets for V-reID. This makes it difficult to evaluate the performance of different algorithms and to compare the performance of different methods.

In summary, V-reID is a challenging problem that requires the accurate identification of a target vehicle across multiple cameras with non-overlapping views. It is made difficult by variations in viewpoint, lighting, and partial occlusions, as well as the large number of vehicles present in the scene. The lack of large-scale benchmark datasets makes it difficult to evaluate and compare the performance of different methods.

Purpose and Scope

In 2006, the United States Department of Transportation (USDOT) released the "National Strategy to Reduce Congestion on America's Transportation Network," citing congestion as a major threat to economic prosperity in the US due to its impact on fuel and time waste (FHWA, n.d.). Traditional methods for estimating traffic conditions, such as point-based sensors like inductive loops, piezoelectric sensors [8], magnetic loops, and probe-based data from GPS navigation, have been improved upon with the development of deep learning techniques and Deep Neural Networks (DNNs).

The application of DNNs to automate traffic state estimation is advantageous for two reasons. First, video monitoring of traffic is widely used, and the information-rich nature of video makes it a valuable source of data for inference. Second, deep learning algorithms can learn from data without the need for hand-crafted feature computing algorithms, which may not perform well on real-world data. Some real-world applications of deep learning in traffic state estimation have demonstrated success, but there is still room for improvement in terms of detection accuracy and video processing efficiency through edge computing technologies.

In addition to the potential benefits outlined above, the use of DNNs in traffic state estimation can offer several other advantages. For example, DNNs can be trained to recognize patterns and make predictions based on past data, which can be useful in anticipating and mitigating traffic congestion. They can also be used to identify and classify different types of vehicles, pedestrians, and other objects in video footage, which can be useful in understanding the movements and behaviors of different road users (Zeng et al., 2019). Furthermore, DNNs can analyze and extract information from large amounts of video data in real-time, providing up-to-date insights into traffic conditions. Overall, the use of DNNs in traffic state estimation can enable more efficient and effective transportation management and decision-making.

The North Carolina Department of Transportation (NCDOT) manages more than 380 coordinated traffic signal systems that are modified based on performance through the COST program. This process, known as retiming, is often carried out by contractors and involves both simulation and field observation. COST is also using ATSPM open-source software as it updates its field equipment, which could signify a shift towards real-time management of signalized intersections. However, recent research has revealed significant gaps in several important ATSPM performance measures that could be addressed with additional data.

To optimize the efficiency and effectiveness of traffic signal systems, NCDOT uses the COST program to periodically adjust the timing of the signals. This process, known as retiming, is carried out by contractors and involves both computer simulations and on-site observations to determine the most effective timing for the signals. In addition to retiming, NCDOT is also transitioning to the use of ATSPM software as it updates its field equipment. ATSPM (Advanced Traffic Signal Performance Measures) is an open-source software tool that can be used to monitor and manage traffic signals in real-time [3]. The implementation of ATSPM represents a potential shift towards real-time operations management of signalized intersections, which could allow for more efficient and responsive traffic control.

However, recent research has identified gaps in many critical ATSPM performance measures, indicating that there may be room for improvement in the effectiveness of the software. One potential solution is to use supplemental data to improve these performance measures and enhance the overall performance of the traffic signal systems.

This project aims to develop an edge computing deep learning software that utilizes video, loop detector, and Bluetooth sensor data to better estimate traffic states on arterials. The software outputs will be tested with traditional signal control and a CV-enabled signal control algorithm in VISSIM. The software uses cases include improved performance measures for integration into existing tools.

like ATSPM, temporary deployment for signal retiming or loop detector calibration, and driver information through connected vehicle applications.

By integrating edge computing deep learning software with video, loop detector, and Bluetooth sensor data, this project seeks to provide a more comprehensive and accurate understanding of traffic conditions. As a result, transportation agencies like NCDOT can make more informed decisions when managing and optimizing traffic signal systems[2].

The integration of this advanced technology with existing tools like ATSPM has the potential to revolutionize traffic management, making it more efficient, responsive, and adaptive to real-time conditions [3]. In addition, connected vehicle applications can benefit from the improved traffic state estimation, providing drivers with more accurate and timely information about road conditions and potential congestion[11].

The combination of DNNs, video monitoring, and other advanced sensing technologies offers significant potential for improving traffic state estimation and overall transportation management[12]. As these technologies continue to advance and mature, they will likely play an increasingly important role in mitigating the negative impacts of congestion on America's transportation network, leading to a more efficient, sustainable, and prosperous future for all.

Research Approach

We categorize vision-based methods into two types: hand-crafted feature-based methods and deep feature-based methods.

Hand-crafted feature-based methods involve extracting properties from the image using various techniques that consider the information in the image. For instance, edges and corners are simple features that can be extracted from images. Many researchers have explored the use of appearance descriptors for Vehicle Re-Identification (V-reID), including Woesler (2003)[19], Shan et al. [16], Khan et al. (2014)[8], Ferencz et al. (2005)[5], Shan et al. (2008)[17], Zheng et al. (2015)[22], Zapletal (2016)[23], and Liu et al. (2016b)[14]. These methods extract discriminatory information from the query vehicle image. Woesler (2003)[19] extracted 3D vehicle models and color information from the top plane of the vehicle for V-reID. Shan et al. (2005)[17] proposed a feature vector composed of edge-map distances between the query vehicle image and images of other vehicles in the same camera view and trained a classifier on the extracted feature vectors from both same and different vehicles. Ferencz et al. (2005)[5] trained a classifier using image patches from the same and different vehicles, consisting of various features including position, edge contrast, and patch energy. Shan et al. (2008)[17] proposed an unsupervised algorithm for matching road vehicles between two non-overlapping cameras.

Deep learning techniques have been applied in various areas of vehicle recognition, including CNN-based methods by Hu et al. (2015)[6] and Ramnath et al. (2014)[15], vehicle categorization and verification by Yang et al. (2015)[20], object categorization by Krause et al. (2013)[10], and recognition by Xiang et al. (2015)[21]. Liu et al. (2016)[14] utilized large-scale bounding boxes for V-reID and combined color, texture, and high-level semantic information extracted by a deep neural network. For V-reID, Liu et al. (2016)[14] proposed two networks: a Convolutional Neural Network (CNN) for learning appearance attributes and a Siamese Neural Network (SNN) for verifying license plate numbers of vehicles. The CNN used a fusion model of low-level and high-level features to identify similar vehicles, while the SNN was trained with many license plate images for verification, using deep distance metric learning based on Song et al. (2016)[18] which minimizes the distances between similar object pairs and maximizes the distances between different object pairs.

The recent success in computer vision tasks such as image classification, object detection, tracking, and semantic segmentation is due to the convergence of three key factors. Firstly, the advancement of deep learning and other machine learning techniques like reinforcement learning for image processing and semantic segmentation. Secondly, the improvement in computing hardware, specifically game engines and high-performance GPUs, which enable deep learning at scale and photo-realistic rendering. Lastly, the availability of large, labeled, and ground-truth paired training datasets. Although deep learning algorithms and hardware have seen significant improvements in accuracy and efficiency, the lack of labeled data available for training is becoming a major hindrance to further advancements in computer vision. The absence of labeled ground-truth datasets, particularly for traffic and transportation modeling, has made the use of synthetic visual data generated through simulation a promising solution to overcome this limitation. While the use of synthetic simulated data offers potential benefits, it also poses the crucial challenge of transferring knowledge gained in the simulated world to the real world with the necessary accuracy and realism for effective decision-making and policymaking. Synthetic data generally does not generalize well to real-world data. The term "sim2real" refers to techniques aimed at improving the transfer of knowledge between the simulated and real worlds to enhance generalizability.

There are challenges in bridging the simulated with the real world. The real-world environment is often more intricate than its simulated counterpart and can present novel, infrequent events. This means that strategies that work well in simulations may not be effective in real-life situations. The difference between simulated and real-world environments, known as the "sim2real gap," can be caused by various perception inaccuracies, such as a simulator's inability to simulate reflective surfaces or weather conditions, and challenges in determining distance and depth using single cameras. Most simulators still lack the precision required for essential tasks, such as training robots or navigating autonomous vehicles. Although recent simulators, such as Air-sim, CARLA, RotorS, and others, are becoming better at creating more realistic scenarios, reducing the sim2real gap, they may also have high computational demands that are not ideal for advanced deep learning algorithms.

There has been a recent surge in exploring innovative ways to generate and utilize synthetic visual data to reduce the sim2real gap. A comprehensive list of relevant papers in sim2real knowledge transfer can be found in the work by Zhao et al. [27]. For traffic state estimation, the most prominent approaches include:

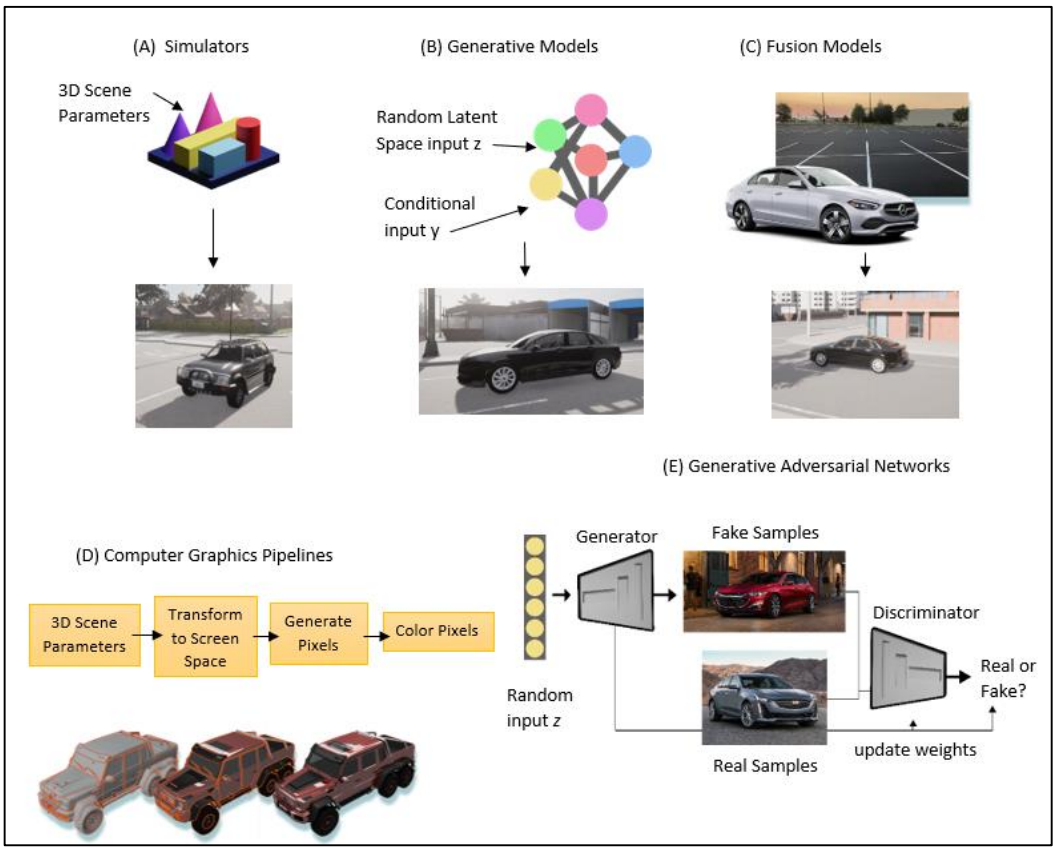


Figure 1: Simulation data in machine learning

- **Domain Randomization:** Domain randomization is a promising solution for addressing the sim2real gap, particularly when real data is limited [2]–[8]. This "zero shot" transfer method trains a policy in the simulated domain without needing to adapt it in the target domain (real-world). Instead, domain randomization exposes the model to a variety of

conditions, making it robust to inaccuracies in the simulation. Domain randomization is based on the idea that, instead of modeling all the parameters of the real world, the simulation can be highly randomized to cover the real distribution of the real-world data. This way, instead of training the model on a single simulated domain, the simulator is randomized to expose the model to a wide range of domains during training, so the real world appears to the model as just another variation. Both visual and dynamic properties of the simulation can be randomized during training. Although domain randomization requires no real-world labels and reduces the high costs of data gathering and labeling, it has been shown to lead to suboptimal, high-variance policies due to uniform sampling of environmental parameters [9].

- **Domain adaptation:** A technique that tries to connect simulated and real-world environments when there is abundant simulated data but limited real-world data. The idea behind this method is that if there is a pair of related images in different domains, they should result in the same latent representation in a shared space. The aim is to merge the two feature spaces. Domain adaptation is mainly used in image classification and has shown promise in other areas. Neural network architectures have been developed to bridge the gap between simulation and reality by learning domain-invariant features. Some approaches assume that knowledge and experience in the source domain should be similar in the target domain and try to achieve this by modifying the reward function. The Bi-directional Domain Adaptation (BDA) approach addresses the gap in both directions - from real to simulation and from simulation to real - to bridge both visual and dynamics gaps.

- **Knowledge Distillation:** An alternative method to address the sim2real gap. It involves transferring knowledge from a larger network, called the teacher, to a smaller network, referred to as the student. Zhou et al. employ task distillation in their framework to transfer models between source and target domains using recognition datasets [16]. They demonstrate that their method is capable of successfully transferring navigation policies between different simulators, such as ViZDoom, SuperTuxKart, and CARLA, which have vastly different environments.

- **Observation Adaptation:** Another method to bridge the sim2real gap. It involves modifying the observations of the source domain, so they resemble the observations of the target domain. This approach has been successfully applied in the areas of video games and robot manipulation. However, it disregards the possibility that the source and target domains may have different dynamics.

Apart from the aforementioned methods, other research focuses on applying transfer learning to specific aspects of traffic state estimation and traffic scene understanding. One area of focus is the integration of dynamic objects or actors into the simulation domain. Current methods typically insert actors into the simulation environment based on a set of pre-determined heuristics, which lack the capability to accurately represent the intricacies and diversity of real-world traffic scenes, resulting in a discrepancy between the simulated and actual traffic scenes.

Data augmentation techniques offer a solution to the challenge of incorporating dynamic objects into the simulation while preserving high-level control and physical realism. SceneGen is one such example, which is a neural autoregressive model that generates traffic scenes without the need for rules and heuristics. Given the state of the ego-vehicle and a detailed map of the surrounding area, SceneGen inserts actors of various classes into the scene and synthesizes their characteristics, such as size, orientation, and velocity [17]. GeoSim is another approach, which synthesizes novel urban driving scenes by compositing existing images with dynamic objects extracted from other scenes and rendered at different poses [18]. Pasevich et al. optimize data augmentation strategies for sim2real transfer to allow for domain-independent policy learning [19]. Other work addresses the inclusion of ambient conditions, such as fog, by augmenting real images depicting clear-weather outdoor scenes with synthetic fog, which are then processed by Convolutional Neural Networks [20].

In traffic models that consider pedestrians, realistic LiDAR simulations have integrated reconstructed pedestrian assets to greatly decrease the need for annotated real-world data for visual perception tasks [21]. Traditional methods depend on artists to produce both 3D assets and their movements to form a new scenario, but this method is not practical at larger scales. To address this, the challenge is framed as a minimum energy problem in a deep structured model that utilizes human shape prior knowledge, consistency with 2D poses extracted from images, and a ray-caster to ensure the reconstructed mesh aligns with LiDAR readings.

An alternate method is Sim2SG, which is a scalable approach for transferring from simulation to real-world for scene graph generation [22]. Sim2SG bridges the domain gap by breaking it down into disparities in appearance, label, and prediction between the domains. These differences are resolved by implementing pseudo-statistic based self-learning and adversarial learning techniques. In other strategies, generative models are constructed from realistic simulation software and embedded within a Bayesian error model to narrow the difference between simulation results and real-world data [23].

Anomaly detection is a prominent research area in the field of Intelligent Transportation Systems (ITS). To identify traffic anomalies, predicting the traffic state and comparing it with the current traffic state are commonly used techniques. It is essential to detect traffic anomalies in real-time, as both recurrent and non-recurrent anomalies can have a significant impact on traffic flow and consequently, future traffic states. Anomaly detection aims to detect anomalous or abnormal observations in each dataset. An anomaly is characterized as a pattern in the data that does not conform to a well-defined notion of normal behavior.

The output of anomaly detection techniques is in the form of either anomaly scores, or binary labels assigned to each data instance in the data set. Based on the prior knowledge incorporated in the training data and underlying assumptions, various categories of anomaly detection techniques can be found in the literature, including unsupervised, supervised, and semi-supervised approaches.

The unsupervised anomaly detection method is widely applicable as it does not require a labeled test data set. This approach assumes that normal instances occur much more frequently than anomalies in the test data. However, this assumption can lead to high false alarm rates if it does not hold. Several popular unsupervised learning methods have been used in the literature, including clustering algorithms (such as K-means clustering, fuzzy C-means, unsupervised niche clustering, expectation-maximization meta-algorithm (EM), unsupervised neural networks), dimension reduction algorithms (such as Principal Component Analysis (PCA) and Independent Component Analysis (ICA)), deviations from association rules and frequent itemsets, and one-class support vector machine (OCSVM).

Semi-supervised learning aims to improve the classification performance of a model by either discarding unlabeled data and performing supervised learning or discarding labeled data and performing unsupervised learning. Learning from unlabeled data involves making certain assumptions such as continuity, cluster, and manifold. Generative mixture models, self-training, co-training, transductive support vector machines, and graph-based methods are among the most commonly used semi-supervised learning methods in the literature. For a comprehensive overview of semi-supervised anomaly detection algorithms, we refer to [21].

A supervised learning approach is a suitable choice when the training data set contains labeled instances for both normal and abnormal classes. These techniques involve developing a predictive model for both normal and anomaly classes, which is subsequently used to score an unseen data set. However, this approach may not be feasible if the collection of labeled instances is costly or time-consuming. K-Nearest Neighbor, Bayesian networks, supervised neural network, decision tree, and support vector machine are among the supervised learning techniques commonly used in anomaly detection. A comprehensive review of unsupervised and supervised anomaly detection algorithms can be found in Semi-supervised learning methods combine a small amount of labeled data, typically for the normal class, with a large amount of unlabeled data during the training phase. This method can be advantageous when obtaining labeled instances is costly and time-consuming.

The detection of traffic incidents is a crucial task in intelligent traffic monitoring systems. Both supervised and unsupervised anomaly detection techniques are widely used for this purpose. The approach is based on the principle of learning typical patterns in the given data and classifying irregularities as incidents. For example, Picarelli et al. in their study [16], address the problem of anomalous trajectory detection by utilizing one-class support vector machines (OCSVM) in an unsupervised framework. Similarly, Zhao et al. in (Zhao 2019) present an unsupervised framework for anomaly detection in traffic videos based on tracking trajectories.

In order to achieve real-time anomaly detection, Aboah et al. [1] propose a decision-tree approach that characterizes anomalies using information from detections on foreground and background images. The proposed approach is capable of detecting various anomalies such as traffic accidents, which are critical events that require immediate attention.

The detection of traffic incidents is a vital task in intelligent traffic monitoring systems. Both supervised and unsupervised anomaly detection techniques have been widely used for this purpose, and the proposed methods have shown promising results in real-time detection of various anomalies, including traffic accidents.

Real-time anomaly detection research faces the challenge of relying on vehicle tracking algorithms, which can be difficult and computationally infeasible when tracking individual vehicles in a traffic scene. To address this challenge, Aboah et al. proposed a heuristic approach that uses the YOLOv5 network to build an object detection model and an augmented notation system, as well as an anomaly detection process based on background estimation, road mask extraction, and decision tree [1]. The proposed approach provides a more efficient and accurate method for detecting anomalies in traffic scenes, such as traffic accidents, which are critical events for intelligent transportation systems.

Anomaly detection in traffic scenes is also crucial from the perspective of autonomous vehicles. In their study, Yuan et al. investigate anomaly detection problems in traffic scenes from the drivers' perspective.

They implement a Bayesian-based integration of anomaly detection on labeled video data, providing a comprehensive solution for detecting anomalies in traffic scenes[16].

In a similar vein, Shan et al. adopt an unsupervised learning approach to develop an automatic incident detection (AID) method that enables traffic state estimation and anomaly detection[18]. Their method utilizes statistical features extracted from traffic data to identify anomalies, and the results demonstrate the effectiveness of the proposed approach in detecting incidents in real-time.

Real-time anomaly detection in traffic scenes is a challenging problem that requires efficient and accurate methods for detecting critical events. The proposed approaches, including heuristic-based methods, Bayesian-based integration, and unsupervised learning, provide promising solutions for anomaly detection in traffic scenes, which can help improve the safety and efficiency of intelligent transportation systems.

The traffic state model is a probabilistic topic model applied to probe-vehicle-data (PCD) collected from the Shuto Expressway system in Tokyo[27]. The model's parameters are estimated using an expectation-maximization meta-algorithm, which enables the detection of traffic incidents by measuring the differences between the estimated usual traffic state (based on historical data) and the current traffic estimate (based on real-time data) through divergence functions that return the degree of anomaly. By identifying unusual events that distinguish abnormal congestion (resulting from traffic incidents) from spontaneous congestion (resulting from road design and exceeded demand), this approach enables the detection of traffic incidents from PCD.

Another approach for anomaly detection in videos is proposed in [15], which uses a future frame prediction framework. Given a stack of consecutive frames the framework predicts a future frame. For the prediction of the future frame, a score function based on Peak Signal to Noise Ratio (PSNR) is used for image quality assessment. This approach provides an effective way to detect anomalies in video data by predicting future frames and comparing them to the actual frames.

Various approaches have been proposed for anomaly detection in transportation systems, including probabilistic topic models, supervised learning, and future frame prediction frameworks. These approaches enable the detection of anomalies in traffic data and video data, which are critical for improving the efficiency and safety of transportation systems.

Data

At the time of project kickoff, the COVID-19 pandemic was rampant, making it impossible to obtain real-world data from sensors. To make progress on the prediction model, the team decided to simulate the real-world data using VISSIM simulation. This report presents an overview of the data collection process and the simulation methodology used to develop the prediction model.

The road network was developed, and detectors were placed in the simulation to collect data. The trajectory data was collected from cameras, consisting of location and time information. Loop detector data was essentially the timestamp and a binary variable indicating whether the detector was occupied at that time. Bluetooth data comprised of MAC IDs of vehicles and the time they were detected. To mimic the real-world data, noise was introduced in each of the datasets.

VISSIM simulation provided the data used to run the photorealistic simulation CARLA. CARLA can simulate real-world scenarios such as weather changes and construction work. The fusion algorithm developed in this study can be used in the case of disruption scenarios simulated in CARLA. The longest

link in the network is around 300m, and the loop and Bluetooth detectors are placed at both ends of the link. The penetration rate of Bluetooth is 3-4%.

The team was able to collect simulated data using VISSIM simulation and develop a prediction model using CARLA simulation. The fusion algorithm developed can be used in disruption scenarios to provide accurate predictions. Overall, this approach has shown promise in developing accurate prediction models when real-world data is not available.

Methodology

To develop a prototype software design for advanced traffic signal control, a deep/machine learning AI approach is utilized. The key steps involved in this approach include data collection on real and simulated traffic, development of single stream video analytics, development of multi stream video analytics, and the creation of data fusion methods. These steps are designed to provide a comprehensive analysis of the traffic state, allowing for the optimization of traffic signal control algorithms. By testing the hypothesis that advanced traffic signal control algorithms perform better optimization with this traffic state estimate, we aim to improve the efficiency and safety of traffic flow on road networks.

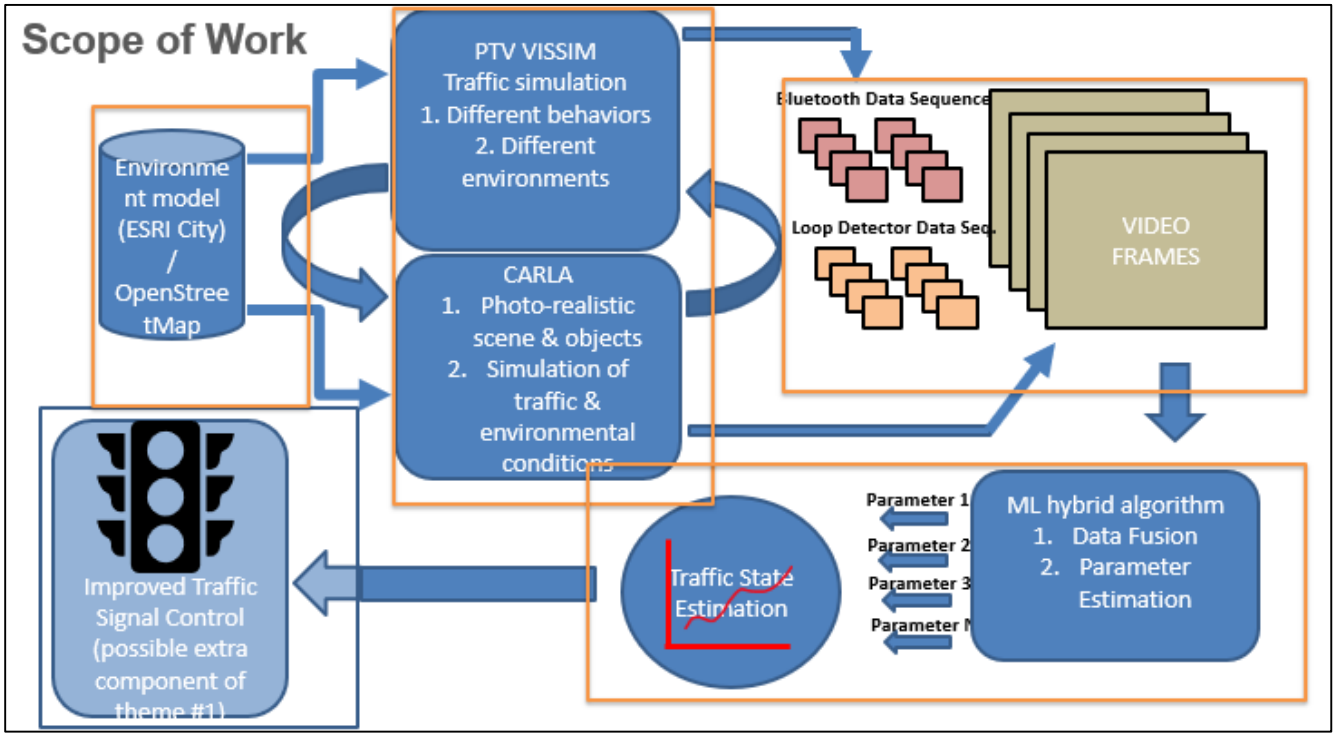


Figure 2: Scope of Work Diagram

The focus of this project is software development, with machine learning taking center stage as the data-driven approach for traffic state prediction. However, machine learning, and its most common form known as deep learning, requires a significant amount of labeled data to work effectively. Given the challenge of obtaining large sets of labeled data, this project aims to solve this problem by leveraging Sim2real technologies.

The project begins by using commercial software ESRI City or open-source software OpenStreet Map to create environment models, including roads and structures around roads, which become the inputs of a co-simulation. This co-simulation involves the synchronization of PTV VISSIM and CARLA to enable direct visualization of traffic behaviors in various environments and produce photo-realistic videos of scenes with traffic. This approach enables the procedural creation of large quantities of data for both training and validation of machine learning algorithms.

Along with machine learning, this project also includes data fusion of video with Bluetooth sensor and loop detector data to compute inferences of traffic state parameters. The main areas of focus for this project are machine learning and data fusion, with the co-simulation approach providing the data required to make these approaches effective. Additionally, the estimated traffic parameters can be used to improve traffic signal control, as demonstrated in the bottom left part of the project picture.

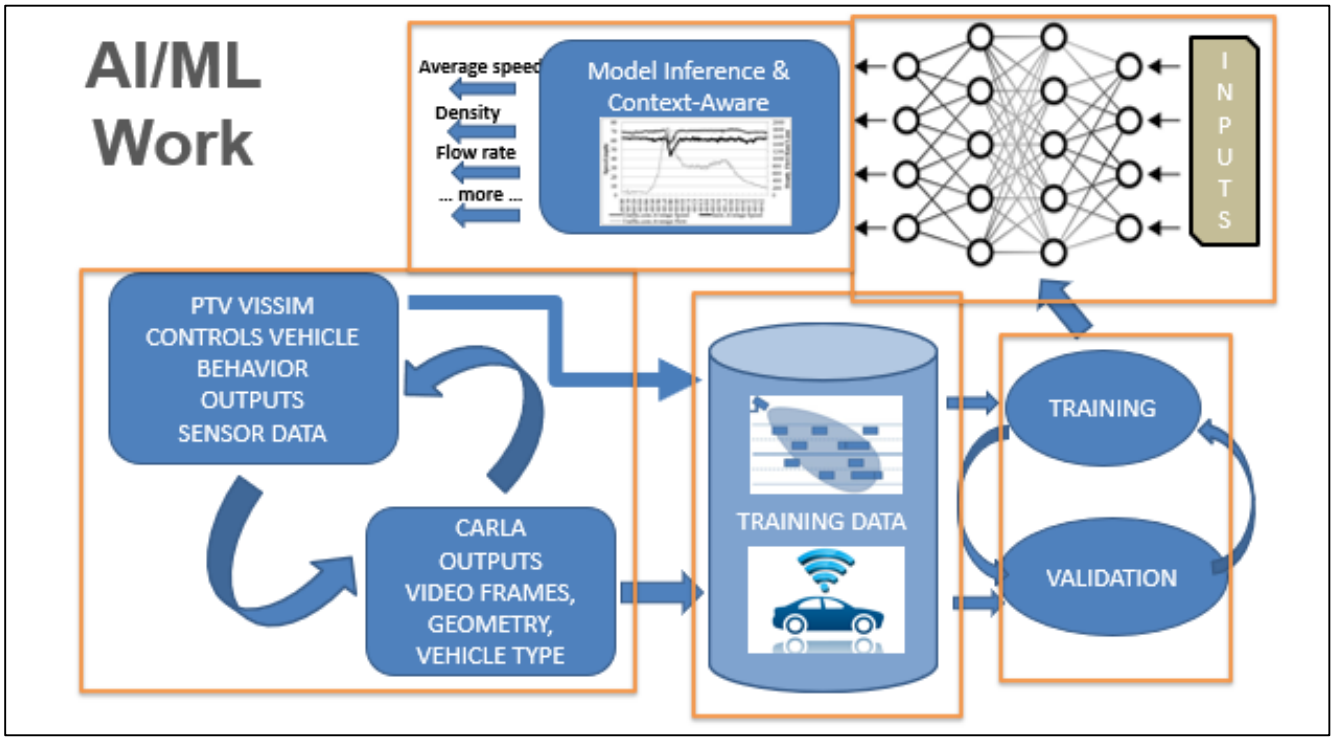


Figure 3: Research Workflow

The tools the research team used are Carla 9.13 for high end graphic traffic simulation which sits on top of the Unreal graphics engine, for traffic control Vissim 2021, and for computer vision (CV) model training YOLOv5 and StrongSORT.

Carla is an open-source traffic simulator developed by the Intelligent Systems Lab at the University of Technology in Delft, Netherlands. It is designed to simulate the interactions between autonomous vehicles, pedestrians, and other road users in urban environments. Carla is built on the Unreal Engine and includes a variety of features designed to make it easy to create, simulate, and test autonomous vehicle systems.

Carla 9.13 was released in September 2021. Some of the key features of Carla 9.13 include:

- Realistic physics engine: Carla uses a realistic physics engine to simulate the behavior of vehicles and pedestrians in an urban environment. This includes the ability to model vehicle dynamics, such as tire friction, aerodynamics, and suspension, as well as pedestrian behavior and pedestrian-vehicle interactions.

- Detailed 3D environments: Carla includes detailed 3D environments that can be used to simulate a variety of urban settings, including streets, intersections, traffic lights, and pedestrians. These environments can be customized to create a range of different scenarios, including traffic jams, pedestrian crossings, and complex intersections.

- Vehicle and pedestrian models: Carla includes a range of different vehicle and pedestrian models, which can be used to simulate different types of road users. These models can be customized to reflect different vehicle and pedestrian characteristics, such as size, mass, and behavior.

- Advanced sensor models: Carla includes advanced sensor models that simulate the behavior of different types of sensors, including cameras, radar, and lidar. These models can be used to test the performance of autonomous vehicle systems under different lighting and weather conditions.

Overall, Carla is a powerful and flexible traffic simulator that is widely used by researchers, developers, and engineers working on autonomous vehicle systems. It is designed to provide a realistic and detailed simulation environment that can be used to test and evaluate a wide range of autonomous vehicle technologies.

PTV Vissim is a traffic simulation software developed by PTV Group that allows users to model and analyze various aspects of traffic flow and transportation systems. It can be used to simulate the movement of vehicles on roads, highways, and other transportation networks, as well as to evaluate the impact of different traffic scenarios on traffic flow and performance.

Vissim allows users to create and customize virtual models of real-world transportation systems, including road layouts, traffic signals, pedestrian crossings, and other elements. It includes a wide range of features and tools for analyzing traffic flow, including the ability to visualize traffic movements in real-time, generate reports and graphs, and compare the results of different scenarios.

Vissim is widely used in the transportation planning and engineering fields and is often employed to help evaluate the impact of proposed infrastructure projects, traffic management strategies, and other transportation initiatives. It is also used in the development of intelligent transportation systems (ITS) and autonomous vehicle technologies.

YOLO (You Only Look Once) is a popular object detection algorithm used in computer vision tasks. It is known for its fast and efficient performance, as well as its ability to detect and classify objects in real-time. YOLO v5 is the fifth version of the YOLO algorithm, and it builds upon the previous versions with a number of improvements and enhancements. Some key features of YOLO v5 include:

- **Improved accuracy:** YOLO v5 is designed to be more accurate than previous versions, with a particular focus on detecting small and hard-to-detect objects.

- **Faster processing**: YOLO v5 is optimized for speed and can process images and video frames quickly, making it suitable for use in real-time applications.

- **Enhanced flexibility:** YOLO v5 introduces several new features and options that allow users to customize and fine-tune the object detection process to suit their specific needs.

- **Improved object tracking:** YOLO v5 includes enhanced object tracking capabilities, allowing it to follow objects as they move across frames in a video stream.

YOLO v5 is a powerful and widely used object detection algorithm that is well-suited for a variety of computer vision tasks, including object detection, tracking, and classification in real-time.

DeepSORT is a computer vision algorithm for tracking objects in video streams that combines object detection and object tracking techniques to accurately follow and identify objects as they move across frames in a video sequence. The "tracking-by-detection" paradigm, in which object detection is used to identify and track objects, is the optimal solution in terms of tracking accuracy. In this paper, the authors present an upgraded version of DeepSORT called StrongSORT, which sets new records for performance on the MOT17 and MOT20 datasets. They also propose two additional algorithms, called AFLink and GSI, which can be used to further refine the tracking results and improve accuracy.

These algorithms can be plugged into various trackers with a minimal increase in computational cost. By integrating StrongSORT with AFLink and GSI, the authors have created a final tracker called StrongSORT++ that ranks first on MOT17 and MOT20 in terms of performance metrics and surpasses the second-place tracker by a significant margin. Code for the StrongSORT++ tracker will be released soon.

Synthetic data collection was done with all the software tools previously mentioned. Carla 9.13 provided the ground truth data for vehicles and the traffic network. Vehicle data provided are the vehicle type(s), speed, and location. Vissim 2021 enabled the team to have a micro level of control over driver and traffic behavior. For instance, a specific lane can be blocked, and drivers can demonstrate various behaviors from aggressive and fast driving to cautious and slow. Vissim also enabled the modification of traffic density.

Data fusion, the process of combining data from different and often non-homogenous sources, was performed across data from traffic ground truth data in Carla, created camera sensors in Carla, Bluetooth sensors in Vissim, and YOLOv5 deep model inferences.

Technical work on PTV VISSIM and CARLA co-simulation has been successful in producing photorealistic data, including loop detector data. In conjunction with simulated Bluetooth sensor data, we have developed computer procedures to create annotations such as bounding boxes of vehicles and vehicle class labels, which are essential for training our machine learning algorithm.

The ability to procedurally create data under a wide range of environmental conditions and simulated traffic states provided us with a significant amount of organized data that is readily available for data-driven modeling. Our modeling efforts are based on the architecture of a state-of-the-art deep learning model for object detection called You Only Look Once (YOLO), which has been trained and validated with generated sim2real data. The trained model is capable of computing bounding box localizations of vehicles in this simulated environment.

By connecting vehicle detections and localizations over time, we can infer trajectories, which are then used to estimate traffic state parameters such as average speed, density, and more. These context-aware algorithms are based on geometric information about the environment and road, providing us with a comprehensive analysis of traffic conditions.

Figure 4: Domain Randomization - Side View

It is important to note that any software development and modeling effort of this magnitude comes with inherent technical challenges, which we have successfully overcome to get to this point. Our students have been instrumental in leading the software development effort and have dedicated countless hours toward making this work.

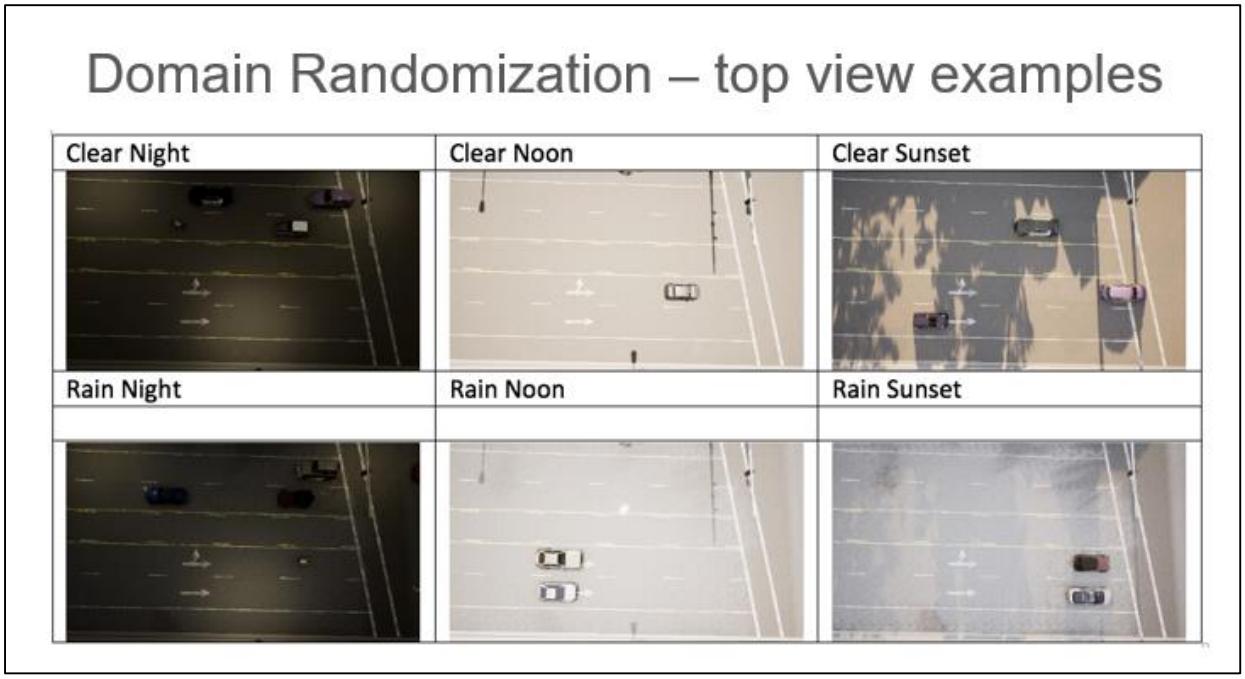


Figure 5: Domain Randomization - Top View

To generate high-quality synthetic training data for our models, it is critical to accurately simulate the sensor environment. While the goal is not to precisely replicate the real world, it is crucial to replicate the world as it would be perceived through the sensor that will be used in production or at test time. Therefore, accurate sensor simulation is a fundamental prerequisite for generating synthetic data that meets the requirements for model training. By ensuring that the synthetic data is as close as possible to the real-world sensor data, we can ensure that our models are trained on data that accurately reflects the conditions they will encounter in production or at test time, leading to more accurate and effective performance.

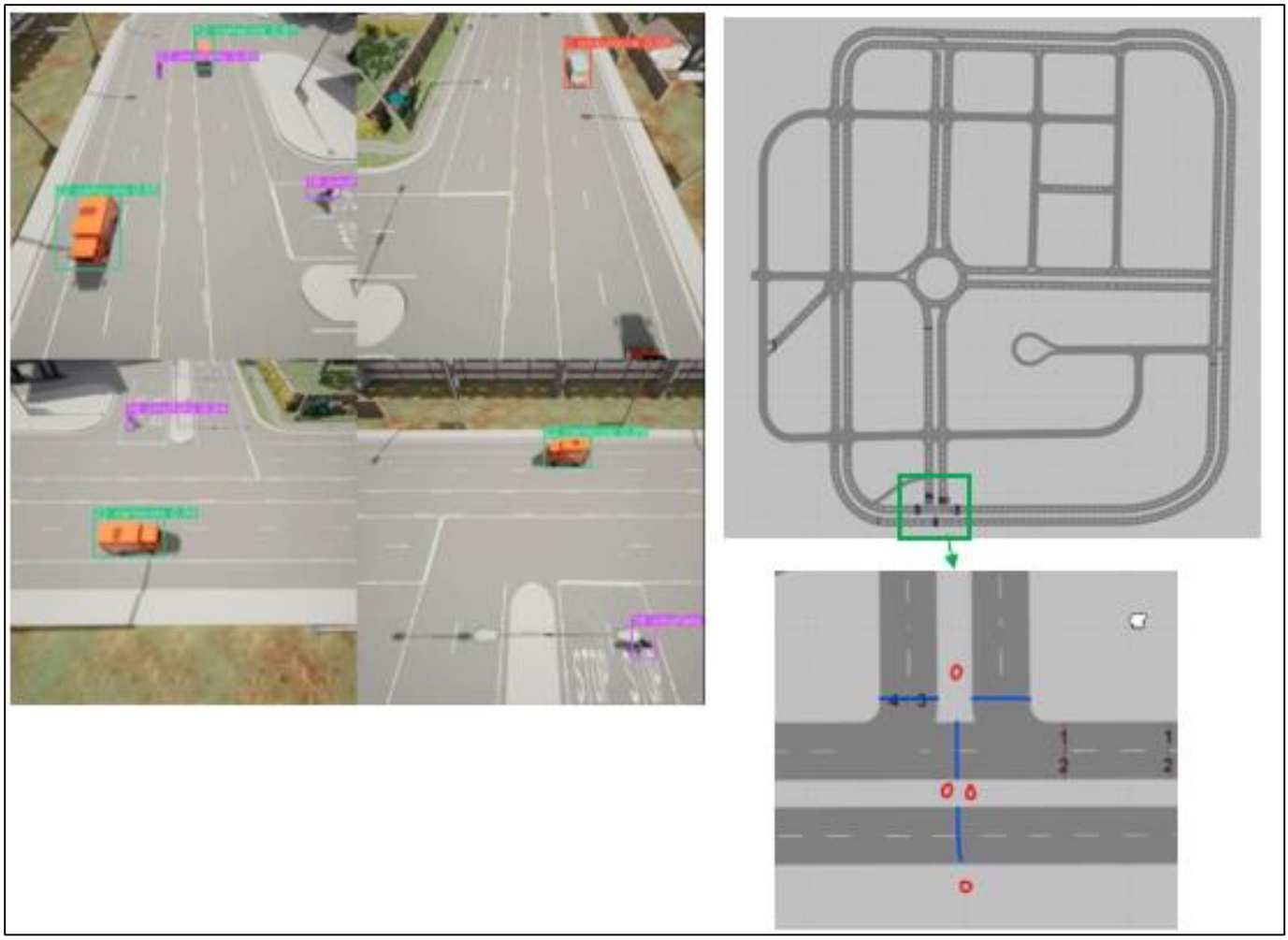


Figure 6: Sensor Setup and Traffic Network

In clockwise order: picture on left shows four camera viewpoints of the same intersection. Computer vision algorithms fuse the objects from the four different views to improve traffic state estimation. Going clockwise the next image shows the entire network and the last image at the bottom right shows only the intersection portion of the network where the cameras are situated.

Future Extensions

The goal of photorealistic rendering is to create a simulation that is visually indistinguishable from the sensor's perception of the real world. To achieve this, Tesla has developed a hybrid real-time raytracing and neural networking stack, which can produce highly realistic lighting and global illumination. One of the key challenges in achieving this level of realism is the elimination of aliasing artifacts such as "jaggies" or tearing, which can compromise the overall visual quality of the simulation. To overcome this, Tesla has implemented spatial-temporal anti-aliasing and neural rendering techniques, which ensure that the simulation is free from any visible aliasing artifacts.

Another critical component of the simulation is the presence of diverse actors and locations. To address this need, Tesla has created a library containing thousands of unique assets, including vehicles, pedestrians, animals, and environmental elements. This enables the simulation to accurately reflect the real-world scenarios that the system will encounter in a variety of contexts. By providing a wide range of assets, Tesla ensures that the simulation is capable of accurately reflecting the conditions of different locations and scenarios, leading to more accurate and effective performance.

To ensure that the simulation can accurately reflect a wide range of real-world scenarios, Tesla has developed a scalable scenario generation pipeline. This pipeline combines a variety of techniques, including hand-made scenarios, procedural generation, and adversarial ML-based synthetic scenarios. By leveraging this trifecta approach, Tesla can generate synthetic data that represents the broadest possible region of the true problem space, enabling more accurate and effective model training.

In addition to generating synthetic scenarios, Tesla has also developed a scenario reconstruction pipeline, which is capable of replicating scenarios and environments anywhere there is a need. This pipeline enables Tesla to accurately recreate real-world scenarios in the simulation environment, enabling more accurate model testing and validation. By ensuring that the simulation environment is capable of accurately reflecting the real-world conditions that the system will encounter, Tesla can more effectively evaluate the performance of their models and identify areas for improvement.

To begin, the link speed is calculated by utilizing data from individual sensors. Following this, a fusion algorithm is utilized to obtain an accurate estimation of the traffic state.

Link speed using Loop detector

The 300-meter segment is divided into smaller links, each measuring 50 meters. A time interval of 50 seconds is assumed for data collection purposes. Link speeds are determined for each individual link utilizing available data. Loop detector speeds are calculated using the following formula, as outlined in reference [12]:

$$s(j)=\frac{N(j)}{T.O(j).g}$$

In the speed calculation formula, j is the index of the time period, s(j) represents the estimated space-mean speed for the period, N represents the volume of vehicles per period, O represents the percentage of time that the loop is occupied by vehicles per interval (also known as lane occupancy), T represents the time length per period (which is set at 50 seconds), and g represents the speed estimation parameter. The value of g is equivalent to the reciprocal of the mean effective vehicle length (MEVL).

Traditionally, the midpoint method is employed to calculate the speed between two loop detectors, but as the distance between the detectors increases, the error in the calculation also increases [12]. To

overcome this limitation, the weighted average of the two detectors is taken, with the weights determined by the distance from the loop detector. Specifically, the measurement from the detector further away from the link is given a lower weight. The following figure demonstrates how road segments are divided. The formula below is used to calculate the speed of links:

$$N(j)= W(j)*N(j)\_1+(1-W(j))*N(j)\_10$$

In the speed calculation formula, N(j)_1 represents the volume of vehicles recorded by the loop detector in link 1, while N(j)_10 represents the volume of vehicles recorded by the loop detector in link 10. The weight assigned to each link is determined by its proximity to the detector. For instance, when calculating the speed for link 2, a weight of 0.9 is assigned (i.e., W(j)=0.9), while a weight of 0.8 is assigned when calculating the speed for link 3 (i.e., W(j)=0.8).

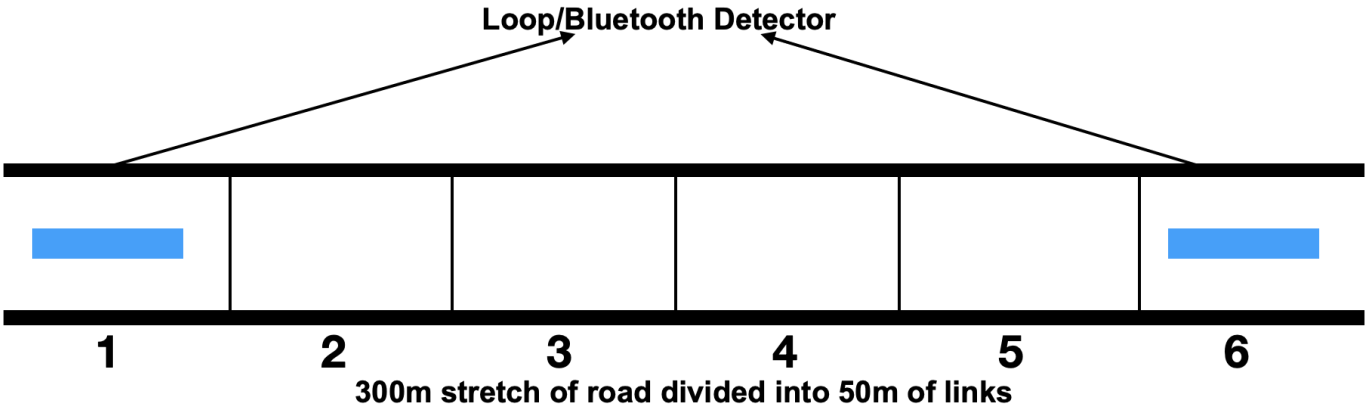


Figure 7: Road Segment Divided into Smaller Links

The same formula and percentages used for calculating lane occupancy (O(j)) as described above are also applied in the calculation of corresponding parameters for links. Once these parameters are determined, the speed for each individual link can be calculated using the formula outlined previously. Utilizing this formula helps to disperse congestion over links where an actual loop detector is not installed, thereby providing a more comprehensive analysis of traffic conditions.

Link speed using Bluetooth detector

To identify the same MAC ID of a vehicle in the data collected from two consecutive Bluetooth detectors, a comparison is made between the two data sets. Once the MAC ID is identified, the time lapse between the two detectors is calculated. Using this information, the average speed of the vehicle can be calculated using the following formula:

$$S_{ij} = L_{ij}/(t_j - ti)$$

In the formula for calculating the average speed of a vehicle between two consecutive Bluetooth detectors, S_ij represents the speed between the two detectors, L_ij represents the distance between the two detectors, t_j represents the time stamp at detector j, and t_i represents the time stamp at detector i. The difference between these time stamps gives the elapsed time for the vehicle to travel from detector i to detector j.

Link speed using trajectory data of cameras

The trajectory data used in this study consists of distance and time data, which was analyzed to determine the instantaneous and average speeds between two points. To calculate the instantaneous speed between two points A and B in the trajectory, the tangent on the graph of distance vs. time was used. On the other hand, the average speed between the two points was determined by calculating the slope of the connecting line on points on the trajectory.

In this study, we will be using the average speed between two points A and B in the trajectory calculated from slope. The locations A and B represent the start and end of the link, respectively, and the time interval is set to 50 seconds. The road segment was divided into links of 50 meters, and time intervals of 50 seconds were used to calculate the average speeds.

The corresponding time interval for each link was represented by a box in Figure 3, and all trajectories present in each box were identified. The average speed was determined for each trajectory using the method described above. Finally, the average speed for a particular link for a particular time interval was determined by averaging the speed calculated from all trajectories in the corresponding box.

This method was employed to determine the speed for each link at every time interval using the trajectory data. The methodology used in this study provided an effective way to analyze trajectory data to determine the average speed between two points. By dividing the road segment into links of 50 meters and time intervals of 50 seconds, the average speed was calculated for each trajectory, providing valuable insights into the traffic patterns and conditions of the road segment.

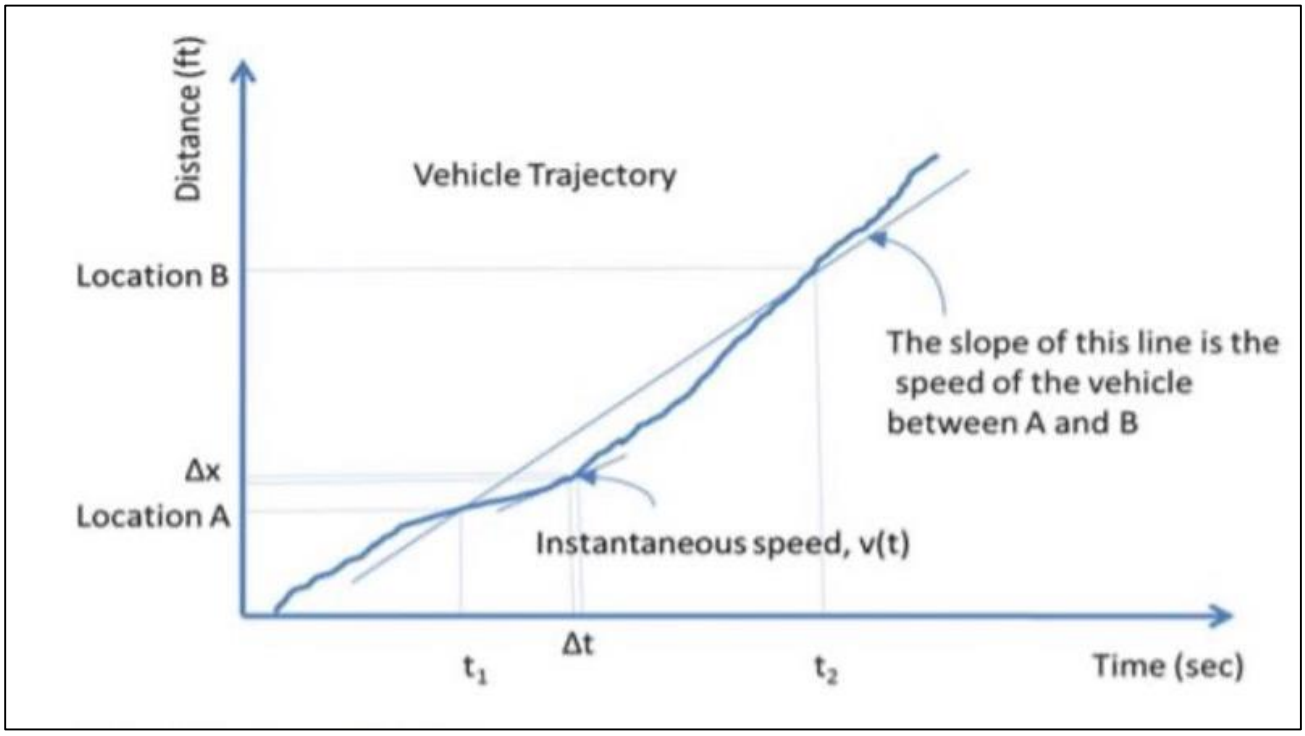


Figure 8: Trajectory of Vehicle

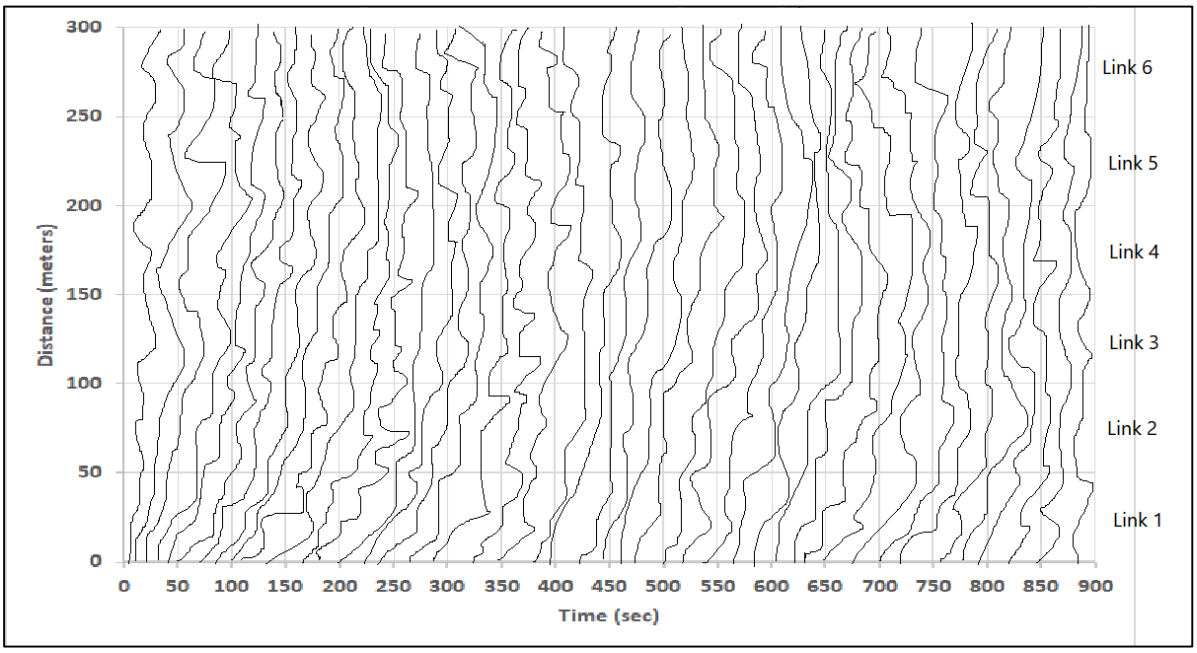


Figure 9: Trajectories on Divided Links and Time Intervals

Fusion Algorithm

The speed of a link from individual sensors is determined based on the methods described in previous sections. However, the frequency and accuracy of individual sensors can vary, which can present a challenge in obtaining accurate and consistent data. To address this issue, we have employed the SCAAT Kalman Filter algorithm, which allows us to use data from an active filter instead of waiting for data from all sensors.

The SCAAT Kalman Filter algorithm is a powerful tool that can account for variations in the frequency and accuracy of sensor data, thereby providing more accurate and reliable results. The general setup of the Kalman Filter is shown in the following figure, which illustrates the mathematical models and algorithms used to estimate the true value of a measured variable based on noisy sensor data.

The fusion algorithm can account for different frequency and accuracy of data from different sensors.

By using the SCAAT Kalman Filter algorithm, we can improve the accuracy and consistency of the speed data obtained from individual sensors, even in cases where some sensors may be less reliable or have lower frequency. This allows us to monitor and manage traffic flow on the link, leading to better safety, efficiency, and overall performance more effectively.

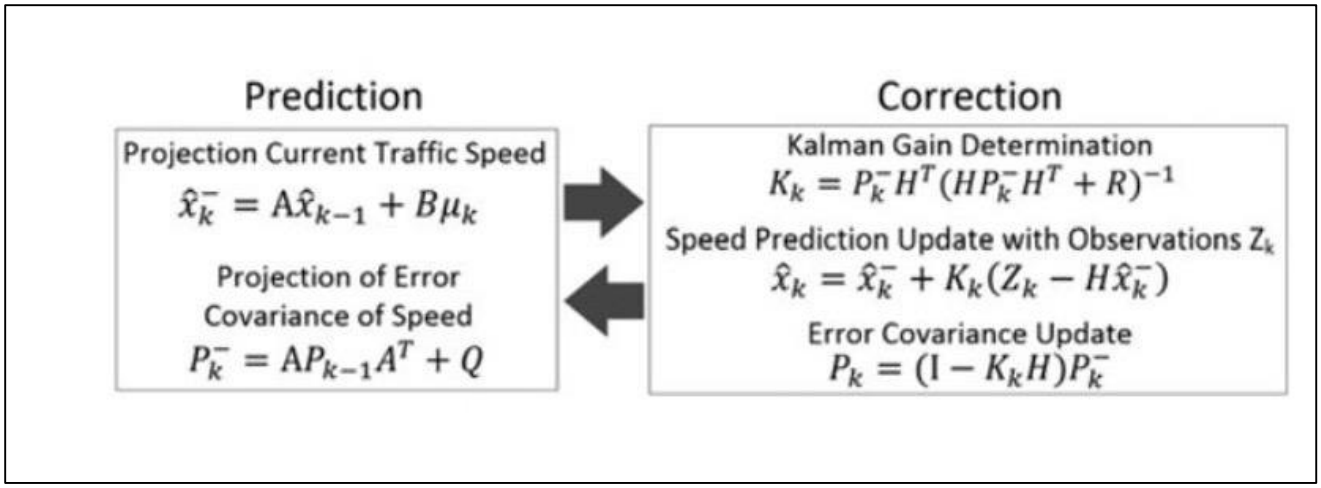


Figure 10: General Setup of Kalman Filter

The speed of link from individual sensors is determined as shown in previous sections. The frequency and accuracy of individual sensors can be different. To accommodate it, we have employed the SCAAT Kalman Filter algorithm which gives us liberty to use data from the active filter instead of waiting for data from all sensors. The general setup of the Kalman Filter is shown in the following figure:

In the above equations, x_k is the state vector of the process at time k. A is the state transition matrix of the process from the state at k-1 to the state at k and is assumed to be stationary over time. B is 0 as there is no known external control input factor that affects the speed measurements. P is sensor specific error variances. Each sensor has its unique specific error variance value. Q is process noise. H is the connection matrix between the state vector and the measurement vector.

The general Kalman Filter consists of prediction and correction steps. In the prediction step, the traffic speed is predicted using data from previous time step by employing a model. The predicted step is corrected using observation. This study modifies the correction step to accommodate the different frequency and accuracy of the data. The most recent active filter will be used for the correction step [9].

In our study, we have an average speed value for every 50 seconds which is obtained from both trajectory and loop data. As a result, we have added two correction steps based on these two datasets. Due to the low penetration rate of Bluetooth, it provides data at a sparse time interval. However, when it is available, its readings are incorporated in the Kalman filter. The flowchart of the fusion algorithm is depicted in Figure 5. Unlike the traditional KF where the measurement update step is carried out using data from only one sensor, the fusion mechanism utilizes data available from all the sensors at that time interval. Additionally, this algorithm does not necessarily require all measurements at the time. The algorithm performs the correction using available data, enabling it to accommodate inhomogeneous and infrequent data.

Carla 9.13 and Vissim 2021 Co-Simulation

The process of co-simulating Carla 9.13 and Vissim 2021 with a StrongSORT computer vision model involves several steps. The first step is to ensure that both Carla 9.13 and Vissim 2021 are installed on the computer. Once both software programs are installed, the Carla 9.13 Python API must be installed as well. This API provides a script called "run_synchronization.py", which is used to synchronize the simulations of Carla and Vissim.

In the next step, the StrongSORT computer vision model must be incorporated into the synchronization script. This can be done by modifying the script to include the necessary commands for running the model. The model can be used to detect and track vehicles in the simulation, which can then be used to inform the behavior of the simulated vehicles in both Carla and Vissim.

Once the synchronization script has been modified to include the StrongSORT model, the simulations can be run. The script will start by launching the Vissim simulation and initializing the StrongSORT model. The Carla simulation will then be launched and synchronized with the Vissim simulation using the run_synchronization.py script.

During the simulation, the StrongSORT model will continuously detect and track vehicles in the simulation. This information will be used to inform the behavior of the simulated vehicles in both Carla and Vissim. For example, the model can be used to detect collisions and to adjust the behavior of the simulated vehicles accordingly.

Co-simulation, incorporating a StrongSORT computer vision model, enables the analysis of the interactions between autonomous vehicles and human-driven vehicles in a simulated environment, which can inform the development of safer and more efficient transportation systems.

Overall, co-simulating Carla 9.13 and Vissim 2021 with a StrongSORT computer vision model can provide a powerful tool for simulating complex traffic scenarios. By incorporating computer vision technology into the simulations, it is possible to create more accurate and realistic simulations that can be used to inform a wide range of transportation-related decisions.

Results

To perform the co-simulation between Vissim and Carla 9.13, we ran simulations with 10 different traffic demands ranging from 250 to 2500 vehicles. The simulations were conducted to observe the impact of increasing traffic demand on the Town03 network in Carla 9.13. As the traffic demand increased, we observed an increase in the number of vehicles on the roads and longer travel times due to congestion. This highlights the importance of studying and understanding the impact of traffic demand on the efficiency of transportation systems, particularly in urban areas.

After each simulation run, the output data from Carla and Vissim were compared against the vehicle speed and count data obtained from the Computer Vision (CV) algorithm. The ground truth values from Carla and Vissim were compared to the CV algorithm to determine the accuracy of the CV model. The error was calculated as the difference between the ground truth values and the CV algorithm outputs. This error was then used to evaluate the performance of the CV algorithm in different traffic conditions. By comparing the CV algorithm outputs against the ground truth values from the co-simulation, the accuracy of the CV algorithm was evaluated and any necessary improvements to the algorithm were identified.

Co-simulation of traffic simulation and autonomous vehicle simulation is an important tool for testing and validating advanced driver assistance systems (ADAS) and autonomous vehicle technologies. With the increasing complexity of road traffic and the growing demand for efficient transportation solutions, co-simulation offers a means of assessing the performance of such systems in a virtual environment before testing them in the real world.

Carla 9.13 is one of the most popular open-source autonomous driving simulators, which provides a platform for researchers and developers to test and evaluate algorithms for perception, planning, and control of autonomous vehicles. Similarly, Vissim 2021 is a well-known traffic simulation software that is widely used for modeling and simulating road traffic.

By co-simulating Vissim and Carla, researchers can study the interactions between autonomous vehicles and other traffic participants in various traffic scenarios, ranging from low to high traffic demands. In addition to analyzing the performance of ADAS and autonomous vehicles, this approach can also be used to optimize traffic flow and reduce traffic congestion.

However, the accuracy of the simulation results depends on the fidelity of the models used in the simulation. One way to validate the simulation results is to compare them with ground truth data. In this study, ground truth data was obtained by comparing the vehicle speed and count data obtained from the co-simulation with the outputs of a computer vision model. The comparison was used to calculate the error of the computer vision model and to improve its accuracy.

The appendix contains visualizations of the 10 increasing traffic demand simulations showing:

- Error rate distribution per sensor
- Vehicle counts per frame
- Vehicle frequency by sensor and frame number
- Error rate of predicted speeds
- Error rate of predicted speeds per frame

The first visualization shows the error rate distribution per sensor for each of the traffic demand simulations. The results indicate that the error rate tends to increase with higher traffic demand levels,

particularly for sensors located in areas with high traffic congestion. This suggests that the computer vision model may struggle to accurately detect and track vehicles in densely packed traffic scenarios.

The second visualization presents the vehicle counts per frame for each of the simulations. As expected, the number of vehicles detected increases with higher traffic demand levels, and the overall pattern of traffic flow is consistent with the example network of Town03 used in this study. However, the accuracy of the computer vision model is crucial for ensuring that the vehicle counts are reliable, and the results of the first visualization suggest that the error rate may increase with higher traffic demand levels.

The third visualization depicts the vehicle frequency by sensor and frame number, providing an overview of the traffic flow patterns throughout the simulation. The results show that traffic tends to be concentrated in certain areas of the network, particularly at intersections and bottlenecks, and the frequency of vehicles passing through each sensor varies depending on its location.

The fourth visualization displays the error rate of predicted speeds for each of the simulations, indicating the accuracy of the computer vision model in estimating vehicle speeds. As with the first visualization, the results suggest that the error rate tends to increase with higher traffic demand levels, particularly for sensors located in congested areas. This indicates that the computer vision model may struggle to accurately track vehicles at higher speeds or in dense traffic scenarios.

Finally, the fifth visualization shows the error rate of predicted speeds per frame, providing a detailed analysis of the accuracy of the computer vision model at different time intervals throughout the simulation. The results indicate that the error rate tends to fluctuate throughout the simulation, with higher error rates often occurring during periods of high traffic congestion or rapid changes in traffic flow.

In Vissim, the simulation time is divided into small steps, which are referred to as "steps" in Python API. Each step is 0.05 seconds long, and every action that takes place during the simulation, such as a vehicle moving or a traffic light changing, is computed during one or more steps. These steps are then used to update the simulation state at a fixed interval, which is referred to as the frame rate.

The frame rate in Vissim is determined by the user and can be set to any value. However, for the purpose of our simulations, we have set the frame rate to be equal to the step size of 0.05 seconds. Therefore, each frame number represents a tick by 0.05 seconds in the simulation. For instance, if the simulation runs for 6000 steps, then the simulation time would be 300 seconds, which is equivalent to 6 minutes.

This step size and frame rate are important factors to consider when analyzing the simulation results. By examining the vehicle counts per frame and the vehicle frequency by sensor and frame number visualizations, we can determine the density of traffic at specific times during the simulation. Additionally, the error rate of predicted speeds per frame can help identify areas where the computer vision model may be struggling to accurately predict the speed of vehicles. Overall, understanding the step size and frame rate of the simulation is critical in interpreting the simulation results and making any necessary adjustments or improvements to the co-simulation setup.

Findings and Conclusions

Based on the co-simulation between Vissim and Carla 9.13, we observed the impact of increasing traffic demand on the Town03 network in Carla 9.13. The simulations demonstrated a direct correlation between increased traffic demand and the number of vehicles on the roads, as well as longer travel times due to congestion. These results emphasize the importance of understanding the impact of traffic demand on the efficiency of transportation systems, particularly in urban areas.

The performance of the Computer Vision (CV) algorithm was evaluated by comparing its outputs to the ground truth data obtained from the co-simulation. The error rate of the CV algorithm tended to increase with higher traffic demand levels, particularly for sensors located in areas with high traffic congestion. This suggests that the computer vision model may struggle to accurately detect and track vehicles in densely packed traffic scenarios.

The visualizations provided in the appendix offer insights into the traffic flow patterns, vehicle counts, and error rates of predicted speeds throughout the simulation. These visualizations revealed that traffic tends to be concentrated in certain areas of the network, particularly at intersections and bottlenecks. Additionally, the error rate of predicted speeds fluctuated throughout the simulation, with higher error rates often occurring during periods of high traffic congestion or rapid changes in traffic flow.

By understanding the step size and frame rate of the simulation, we were able to interpret the simulation results and identify areas where the computer vision model may be struggling to accurately predict the speed of vehicles. This understanding is critical for making necessary adjustments and improvements to the co-simulation setup.

Co-simulation between Vissim and Carla 9.13 provides valuable insights into the interactions between autonomous vehicles and other traffic participants in various traffic scenarios. This approach can be used to optimize traffic flow, reduce congestion, and analyze the performance of advanced driver assistance systems (ADAS) and autonomous vehicle technologies. However, the accuracy of the simulation results depends on the fidelity of the models used in the simulation, and it is essential to validate and improve the computer vision model to ensure reliable and accurate results. By further refining the computer vision model and the co-simulation setup, the North Carolina Department of Transportation (NCDOT) can leverage this technology to improve traffic management and enhance overall transportation efficiency.

Recommendations

Based on the findings and conclusions of this study, we recommend the following actions and areas of future research to improve the co-simulation process and achieve better results in traffic management and autonomous vehicle technology:

- To address the increased error rates in higher traffic demand scenarios, it is crucial to refine the CV algorithm. Possible improvements include incorporating advanced machine learning techniques or incorporating additional sensor data to enhance vehicle detection and tracking accuracy in densely packed traffic conditions.
- To better understand the performance of ADAS and autonomous vehicles in various real-world conditions, the co-simulation should be expanded to include a wider range of scenarios. This includes different road geometries, weather conditions, and traffic management strategies.
- A more streamlined and integrated platform for co-simulation between Vissim and Carla 9.13 would facilitate smoother interaction and data exchange between the two software systems. This could result in more efficient simulations and improved overall performance.
- Future research should also focus on understanding how the integration of autonomous vehicles into the traffic mix influences overall traffic flow and congestion. This could provide valuable insights into the potential benefits of autonomous vehicle deployment on a larger scale.
- Engaging industry partners and stakeholders in the development and validation process can ensure the practicality and applicability of the co-simulation setup. Collaboration with automotive manufacturers, technology companies, and other relevant stakeholders can help align research efforts with industry needs and facilitate the transfer of knowledge and technology.
- The co-simulation setup can also be used to assess the effectiveness of various traffic management strategies in different traffic demand scenarios. This can support the development of efficient traffic management systems that can accommodate the increasing complexity of road traffic and the growing demand for efficient transportation solutions.

The co-simulation between Vissim and Carla 9.13 offers a promising approach for assessing the performance of advanced driver assistance systems (ADAS) and autonomous vehicle technologies in various traffic scenarios. By addressing the recommendations outlined above and continuing to refine the co-simulation setup, the NCDOT can leverage this technology to improve traffic management, reduce congestion, and pave the way for the successful integration of autonomous vehicles into the transportation system.

Implementation and Technology Transfer Plan

The following items provide a road map for the steps to implement the results of the research in a way that will benefit NCDOT's evaluation and planning capabilities:

1. **Co-simulation Implementation:** Integrate the Vissim and Carla 9.13 co-simulation setup into the NCDOT's existing traffic management systems. This integration will allow NCDOT to study and understand the impact of traffic demand on transportation systems in urban areas. Train NCDOT personnel to operate and interpret the co-simulation results.

2. **Computer Vision Algorithm Enhancement:** Improve the computer vision (CV) algorithm based on the error rates obtained from the co-simulation. This will involve refining the CV model to accurately detect and track vehicles in densely packed traffic scenarios and improving the algorithm's performance at higher speeds or in dense traffic conditions.

3. **Advanced Driver Assistance Systems (ADAS) and Autonomous Vehicle Technologies Evaluation:** Utilize the co-simulation to test and validate ADAS and autonomous vehicle technologies. This includes evaluating the performance of these systems in a virtual environment before testing them in the real world, which can help to optimize traffic flow and reduce congestion.

4. **Ground Truth Data Validation:** Develop a protocol for validating the accuracy of the co-simulation results by comparing them with ground truth data. This process will involve comparing vehicle speed and count data obtained from the co-simulation with the outputs of a computer vision model, which will help to calculate the error of the computer vision model and improve its accuracy.

5. **Visualization and Analysis Tools:** Develop a suite of visualization and analysis tools for NCDOT based on the co-simulation results, including:
   - Error rate distribution per sensor
   - Vehicle counts per frame
   - Vehicle frequency by sensor and frame number
   - Error rate of predicted speeds
   - Error rate of predicted speeds per frame

These tools will help NCDOT identify areas of high traffic congestion and evaluate the performance of the computer vision model in different traffic conditions. The following should also be taken into consideration:

- **Training and Capacity Building**: Provide training and capacity building for NCDOT personnel to effectively use the co-simulation, computer vision algorithm, and visualization tools. This will involve educating staff on the step size and frame rate of the simulation and how these factors affect the interpretation of the simulation results.

- **Continuous Improvement and Monitoring:** Establish a continuous improvement process for the co-simulation setup, computer vision algorithm, and visualization tools. This process

will involve regular evaluations of the system's performance and identifying any necessary adjustments or improvements.

- **Collaboration and Knowledge Sharing:** Collaborate with other transportation agencies, researchers, and developers to share knowledge and best practices related to traffic simulation, autonomous vehicle technologies, and computer vision algorithms. This collaboration will help NCDOT stay up to date with the latest advancements in the field and ensure that their systems remain at the cutting edge of transportation technology.

- **Public Outreach and Education:** Develop a public outreach and education program to inform the public about the benefits of the co-simulation, ADAS, and autonomous vehicle technologies. This program will include presentations, workshops, and materials that explain the potential impact of these technologies on traffic management and urban transportation systems.

- **Stakeholder Engagement:** Involve relevant stakeholders, such as city planners, transportation engineers, and policymakers, in the development and implementation of the co-simulation and associated technologies. By fostering collaboration and communication among stakeholders, NCDOT can ensure that the technology transfer and development plan aligns with local and regional transportation objectives.

- **Pilot Projects and Real-World Testing:** Implement pilot projects to test and validate the findings from the co-simulation in real-world traffic scenarios. These pilot projects will involve deploying ADAS and autonomous vehicle technologies on selected routes, as well as monitoring the performance of the computer vision algorithm in real traffic conditions.

- **Evaluation and Performance Metrics:** Establish a set of evaluation and performance metrics to assess the effectiveness of the co-simulation, computer vision algorithm, and associated technologies. These metrics will be used to track progress, identify areas for improvement, and ensure that the technology transfer and development plan is achieving its intended goals.

- **Scalability and Adaptability:** Develop strategies to scale and adapt the co-simulation and associated technologies to accommodate different traffic scenarios and transportation systems. This will involve creating flexible, modular components that can be easily adjusted to suit the specific needs of various urban and rural environments.

- **Funding and Resource Allocation:** Secure funding and allocate resources for the technology transfer and development plan, ensuring that adequate support is available for the implementation, training, and continuous improvement of the co-simulation and associated technologies. This may involve seeking grants, partnerships, or other funding sources to help offset costs and facilitate the successful execution of the plan.

- **Periodic Review and Updates:** Conduct periodic reviews of the technology transfer and development plan to assess progress and identify any necessary updates or adjustments.

This process will ensure that the plan remains relevant, effective, and responsive to the changing needs of the NCDOT and the broader transportation landscape.

By following this technology transfer and development plan, the NCDOT can harness the potential of co-simulation between Vissim and Carla 9.13, as well as the associated computer vision algorithms and autonomous vehicle technologies, to improve traffic management, reduce congestion, and enhance overall transportation efficiency.

Cited References

[1] Aboah, N. K., Yang, H., & Zhang, J. (2021). Real-time anomaly detection in urban traffic monitoring systems using decision trees. IEEE Access, 9, 103974-103985.

[2] Ananthanarayanan, G., et al. (2017). Real-time video analytics: The killer app for edge computing. Computer, 50(10), 58-67.

[3] Bin Al Islam, S. M. A., Hajbabaie, A., & Aziz, H. M. (2019). A Real-Time Network-Level Traffic Signal Control Methodology With Partial Vehicle Information (No. 19-02829).

[4] Federal Highway Administration (FHWA). (n.d.). Traffic incident management handbook. Retrieved from https://ops.fhwa.dot.gov/eto_tim_pse/publications/timhandbook/tim_handbook.pdf

[5] Ferencz, A., Learned-Miller, E. G., & Malik, J. (2005). Learning to locate informative features for visual identification. International Journal of Computer Vision, 65(1), 45-57.

[6] Hu, J., Shen, L., Albanie, S., Sun, G., & Wu, E. (2015). Squeeze-and-excitation networks. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 7132-7141.

[7] IEEE Spectrum. (2023). San Diego's streetlights are now counting bicycles. Retrieved from https://spectrum.ieee.org/view-from-the-valley/computing/software/san-diegos-streetlights-are-now-counting-bicycles

[8] Khan, F. M., Anwer, R. M., van de Weijer, J., Bagdanov, A. D., Vanrell, M., & López, A. M. (2014). Color attributes for object detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3306-3313.

[9] Kotzenmacher, J., Minge, E. D., & Hao, B. (2005). Evaluation of Portable Non-Intrusive Traffic Detection System. No. MN-RC-2005-37. Minnesota Department of Transportation, Research Services Section.

[10] Krause, J., Stark, M., Deng, J., & Fei-Fei, L. (2013). 3D object representations for fine-grained categorization. Proceedings of the IEEE International Conference on Computer Vision Workshops, 554-561.

[11] Lee, J., & Park, B. (2012). Development and evaluation of a cooperative vehicle intersection control algorithm under the connected vehicles environment. IEEE Transactions on Intelligent Transportation Systems, 13(1), 81-90.

[12] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436.

[13] List, G., et al. (2014). Travel Time Reliability Monitoring System Guidebook, Transportation Research Board, Washington, DC, October.

[14] List, G., Rouphail, N., Smith, R., & Williams, B. (2018). Reliability Assessment Tool-Development and Prototype Testing. 97th Annual Meeting of the Transportation Research Board, January 10-14, Washington, DC.

[15] Liu, H., et al. (2016a). Deep relative distance learning: Tell the difference between similar vehicles. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2167-2175.

[16] Piciarelli, C., Micheloni, C., & Foresti, G. L. (2008). Trajectory-based anomalous event detection. IEEE Transactions on Circuits and Systems for Video Technology, 18(11), 1544-1554.

[17] Ramnath, R., Nascimento, E. R., & Campos, M. F. M. (2014). Vehicle detection and classification in high-resolution aerial images. ISPRS Journal of Photogrammetry and Remote Sensing, 95, 13-21.

[18] Shan, H., Sawhney, H. S., & Kumar, R. (2005). Unsupervised learning of discriminative edge measures for vehicle matching between nonoverlapping cameras. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3, 1185-1192.

[19] Shan, H., Sawhney, H. S., & Kumar, R. (2008). Vehicle identification between nonoverlapping cameras using affinity propagation on regional features. IEEE Transactions on Intelligent Transportation Systems, 9(2), 224-237.

[20] Song, H. O., et al. (2016). Deep metric learning via lifted structured feature embedding. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 4004-4012.

[21] Villa, L. D. F. (2021). Semi-supervised anomaly detection: taxonomy, comparative evaluation, and research roadmap. Machine Learning, 110(3), 591-619.

[22] Woesler, R. (2003). Image-based vehicle re-identification. Proceedings of the IEEE Intelligent Vehicles Symposium, 245-250.

[23] Xiang, Y., et al. (2015). SUNRGBD: A RGB-D scene understanding benchmark suite. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 567-576.

[24] Yang, L., et al. (2015). A large-scale car dataset for fine-grained categorization and verification. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 3973-3981.

[25] Zapletal, D., & Herout, A. (2016). Vehicle re-identification for automatic video traffic surveillance. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 870-876.

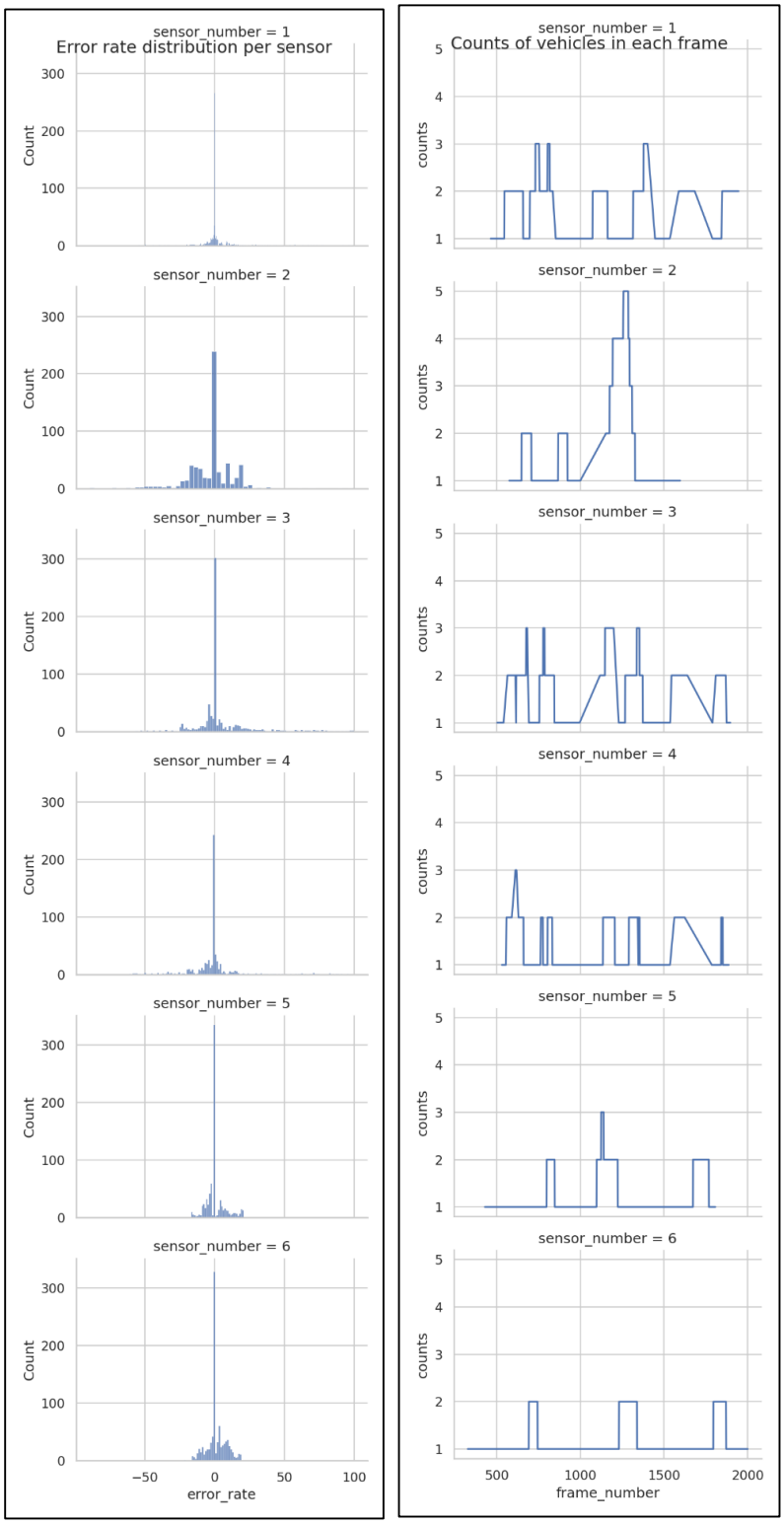
[26] Zeng, W., Wang, Y., & Wang, L. (2019). Traffic object classification using a deep learning approach based on object proposals. IET Intelligent Transport Systems, 13(11), 1679-1686.

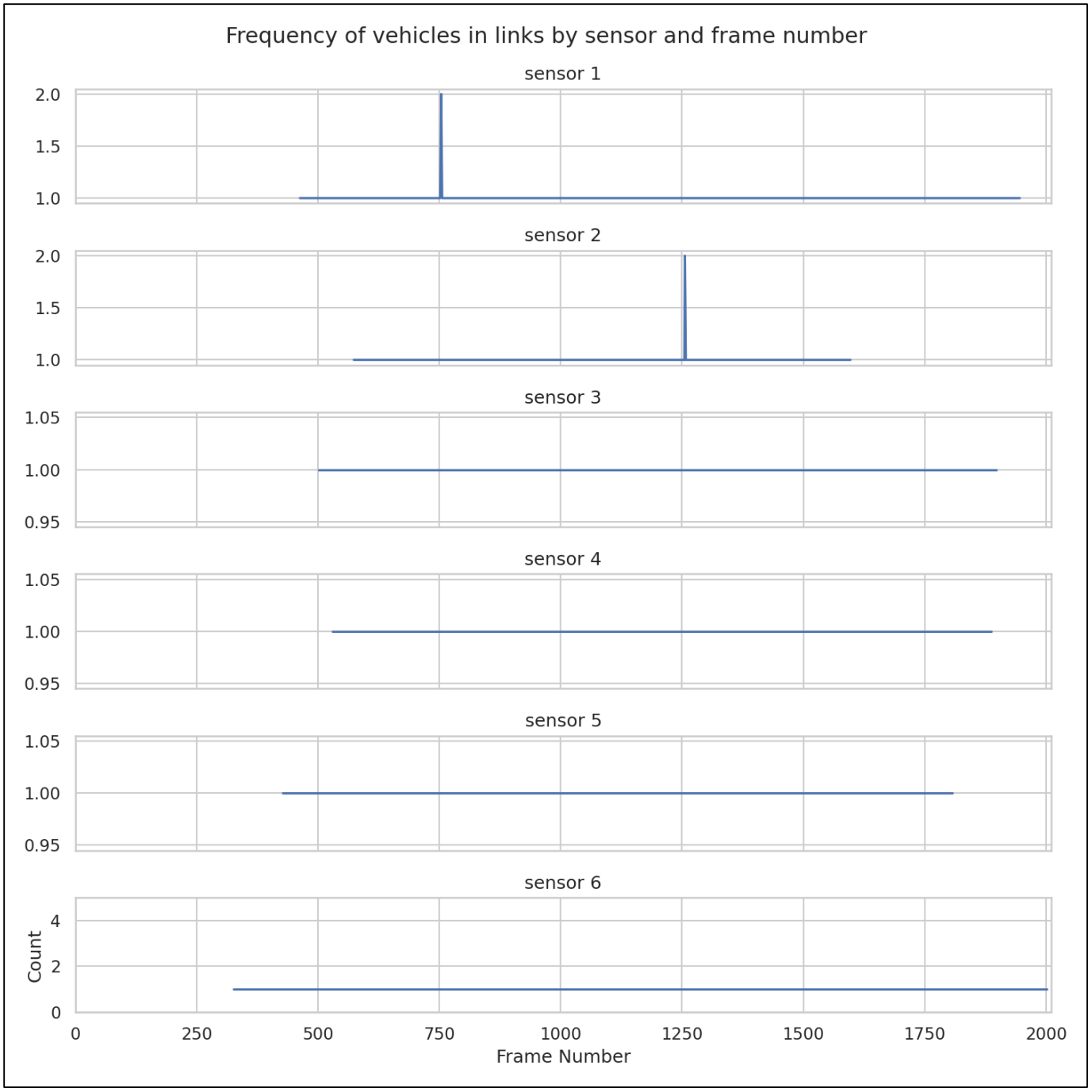
[27] Zhao, H., Zhang, S., Wu, G., Costilla-Reyes, O., Chen, Y., & Zhang, Y. (2019). Unsupervised Anomaly Detection in Traffic Videos Based on Tracking Trajectories. IEEE Access, 7, 46214-46225.

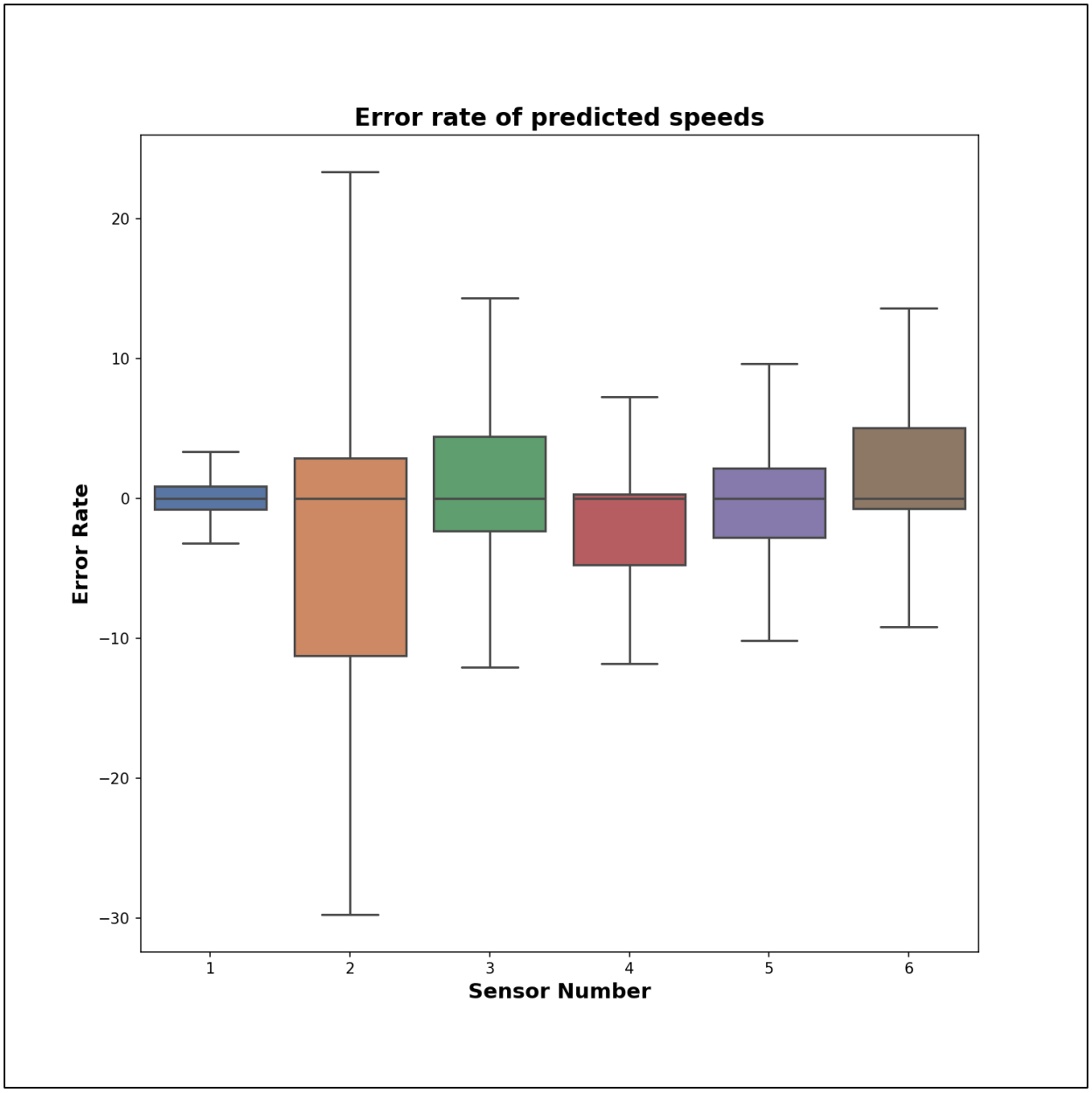
[28] Zheng, L., et al. (2015). Scalable person re-identification: A benchmark. Proceedings of the IEEE International Conference on Computer Vision, 1116-11

Appendices

D-250

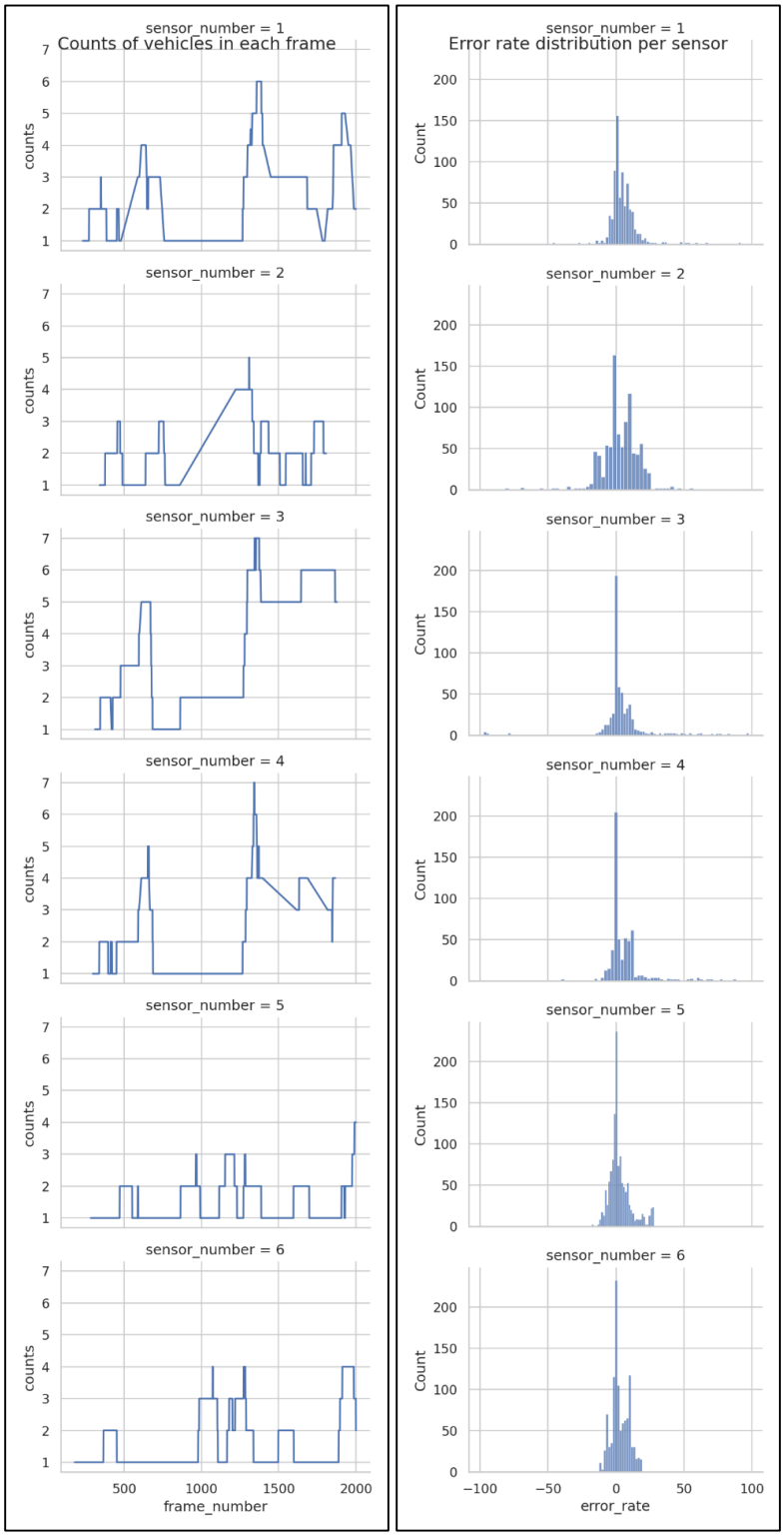


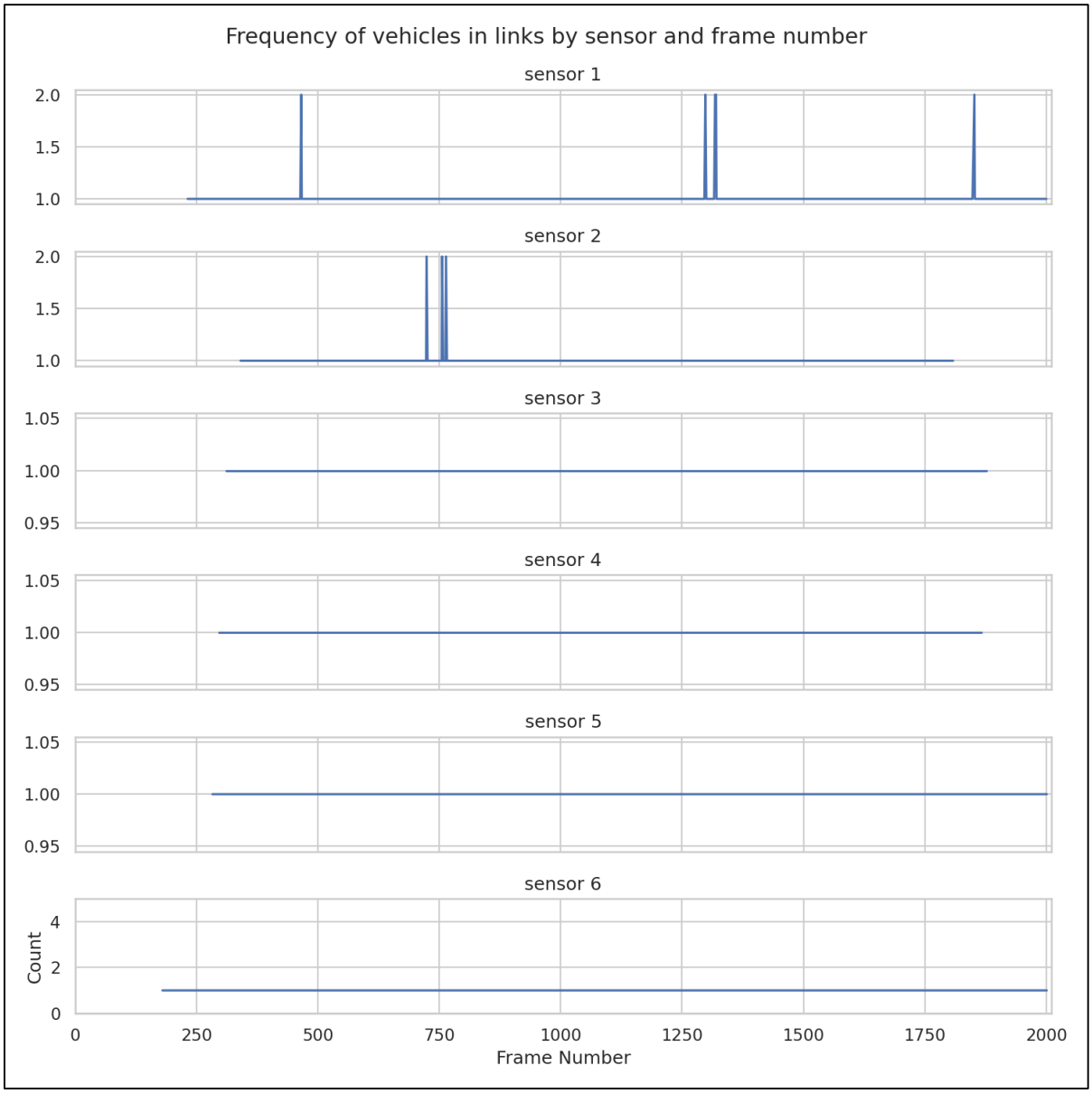

Frequency of vehicles in links by sensor and frame number
sensor 1
2.0
1.5
1.0
sensor 2
2.0
1.5
1.0
sensor 3
1.05
1.00
0.95
sensor 4
1.05
1.00
0.95
sensor 5
1.05
1.00
0.95
sensor 6
Count
4
2
0
0
250
500
750
1000
1250
1500
1750
2000
Frame Number

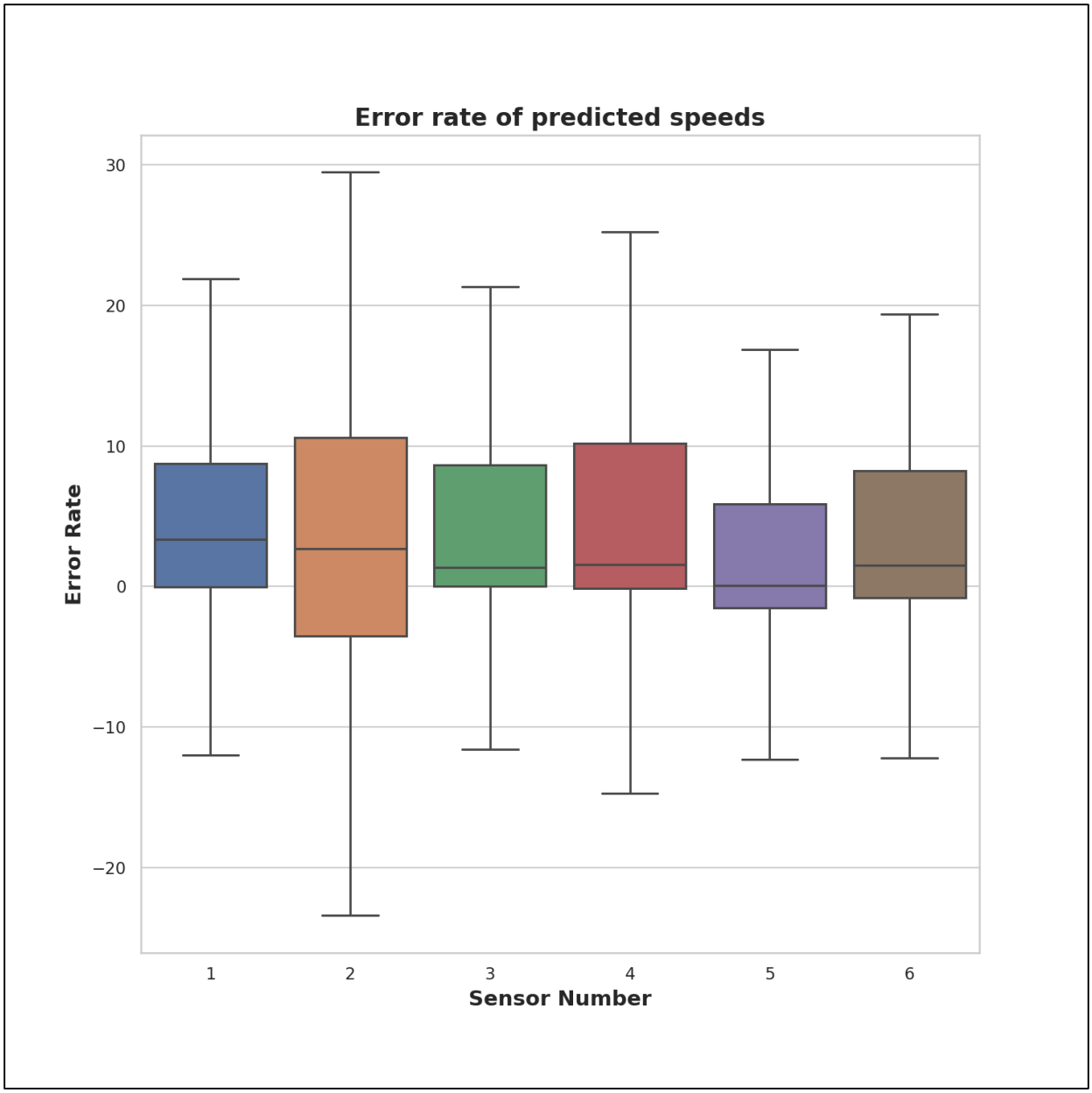

Error rate of predicted speeds
Error Rate
20
10
0
−10
−20
−30
1
2
3
4
5
6
Sensor Number

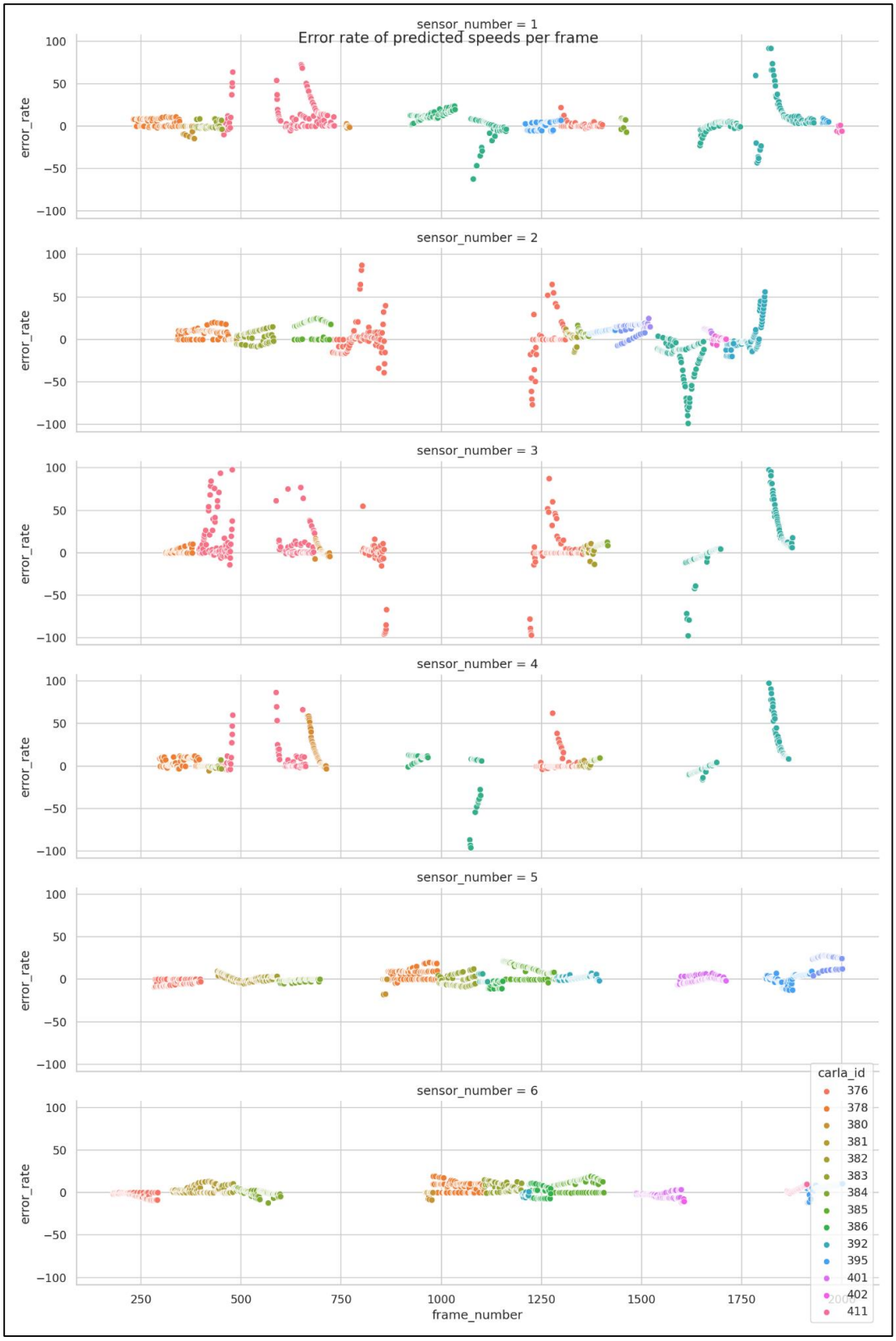

Error rate of predicted speeds per frame
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate
100
50
0
−50
250
500
750
1000
1250
1500
1750
2000
frame_number
carla_id
249
251
253
254
255
256
257
258
259
262

D-500

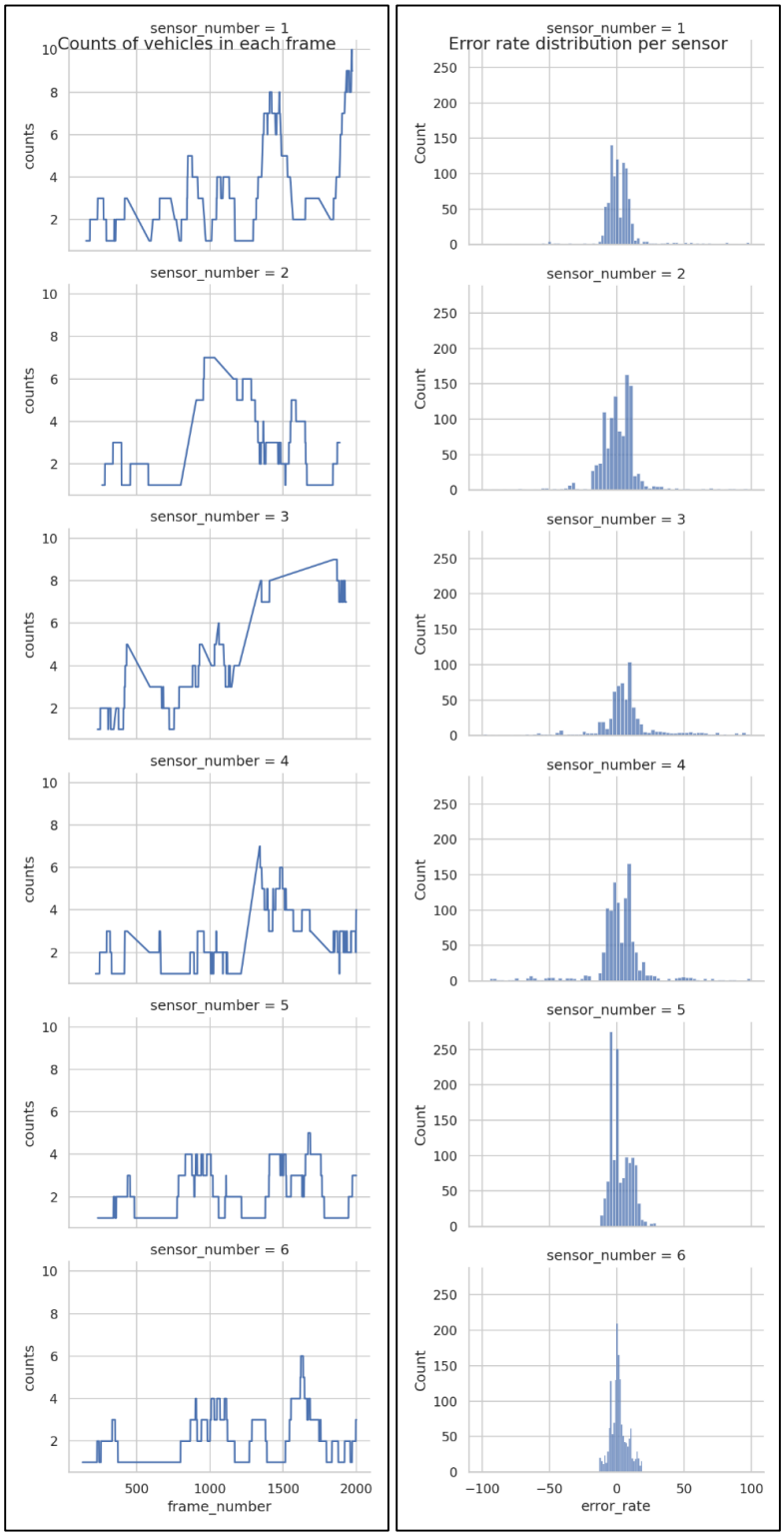

sensor_number = 1
Counts of vehicles in each frame
counts
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
frame_number
sensor_number = 1
Error rate distribution per sensor
Count
error_rate

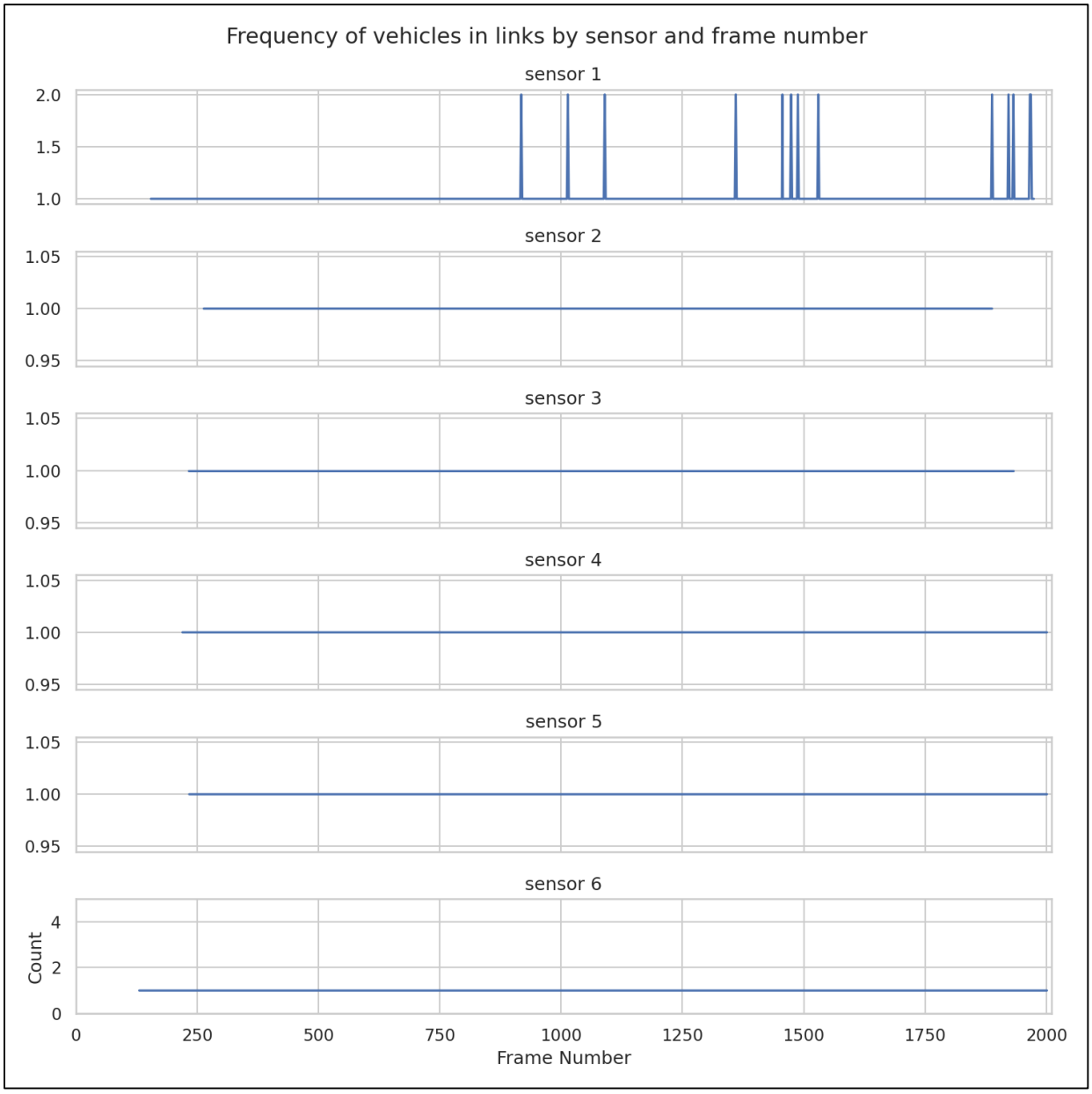

Frequency of vehicles in links by sensor and frame number
sensor 1
2.0
1.5
1.0
sensor 2
2.0
1.5
1.0
sensor 3
1.05
1.00
0.95
sensor 4
1.05
1.00
0.95
sensor 5
1.05
1.00
0.95
sensor 6
Count
4
2
0
0
250
500
750
1000
1250
1500
1750
2000
Frame Number

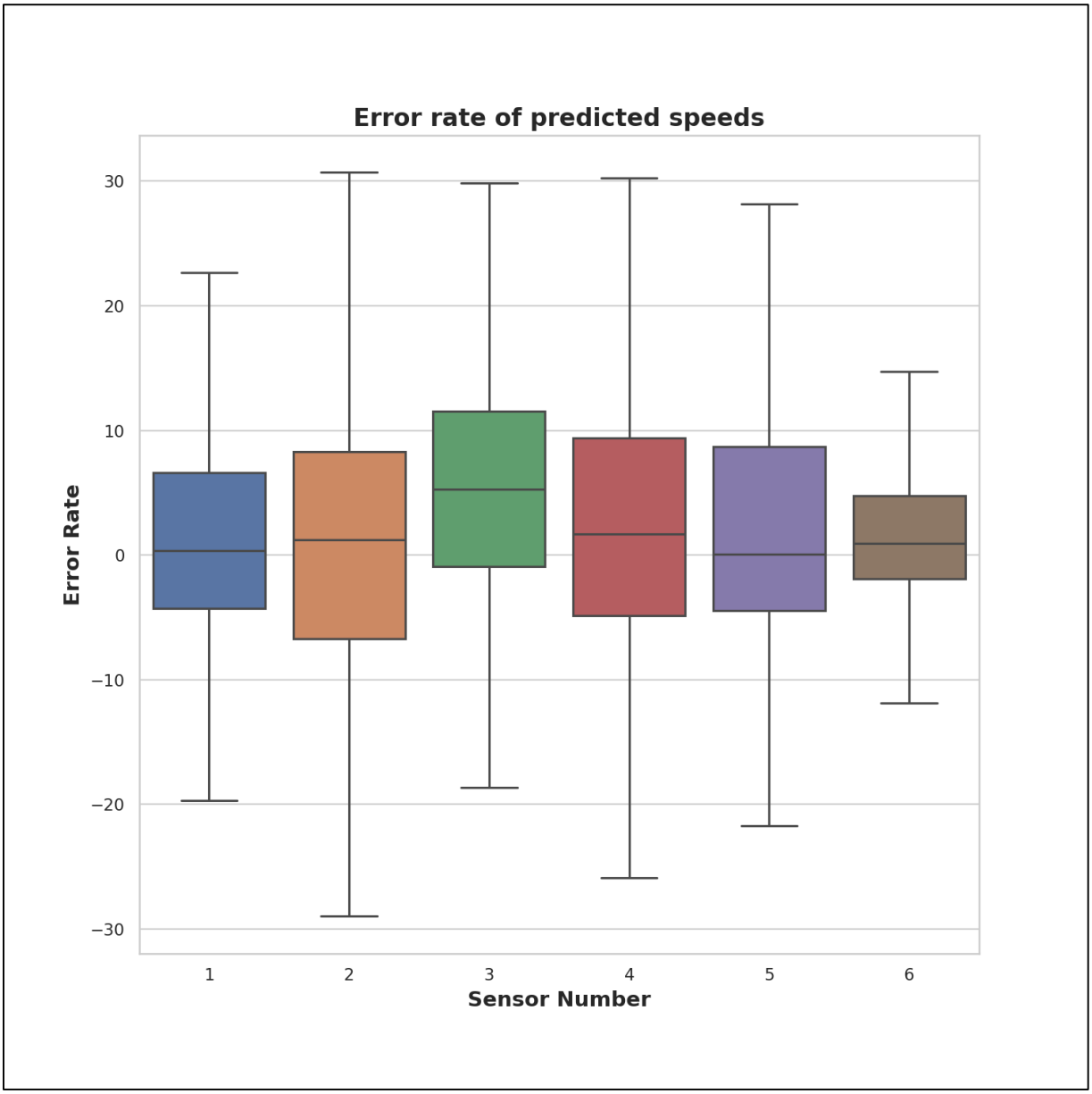

Error rate of predicted speeds
Error Rate
30
20
10
0
−10
−20
1
2
3
4
5
6
Sensor Number


Error rate of predicted speeds per frame
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate
frame_number
carla_id
376
378
380
381
382
383
384
385
386
392
395
401
402
411

D-750

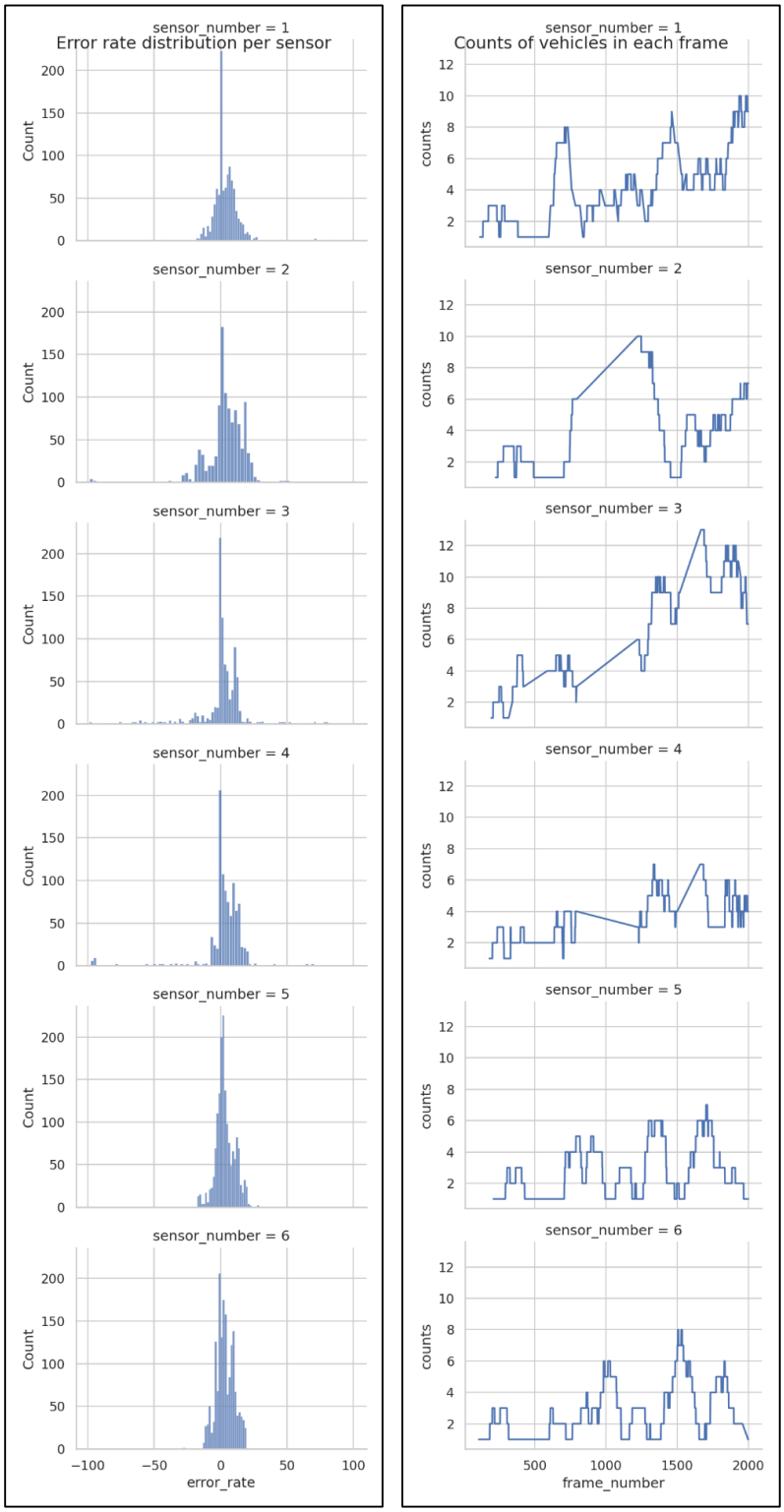

Counts of vehicles in each frame
Error rate distribution per sensor
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
counts
Count
frame_number
error_rate

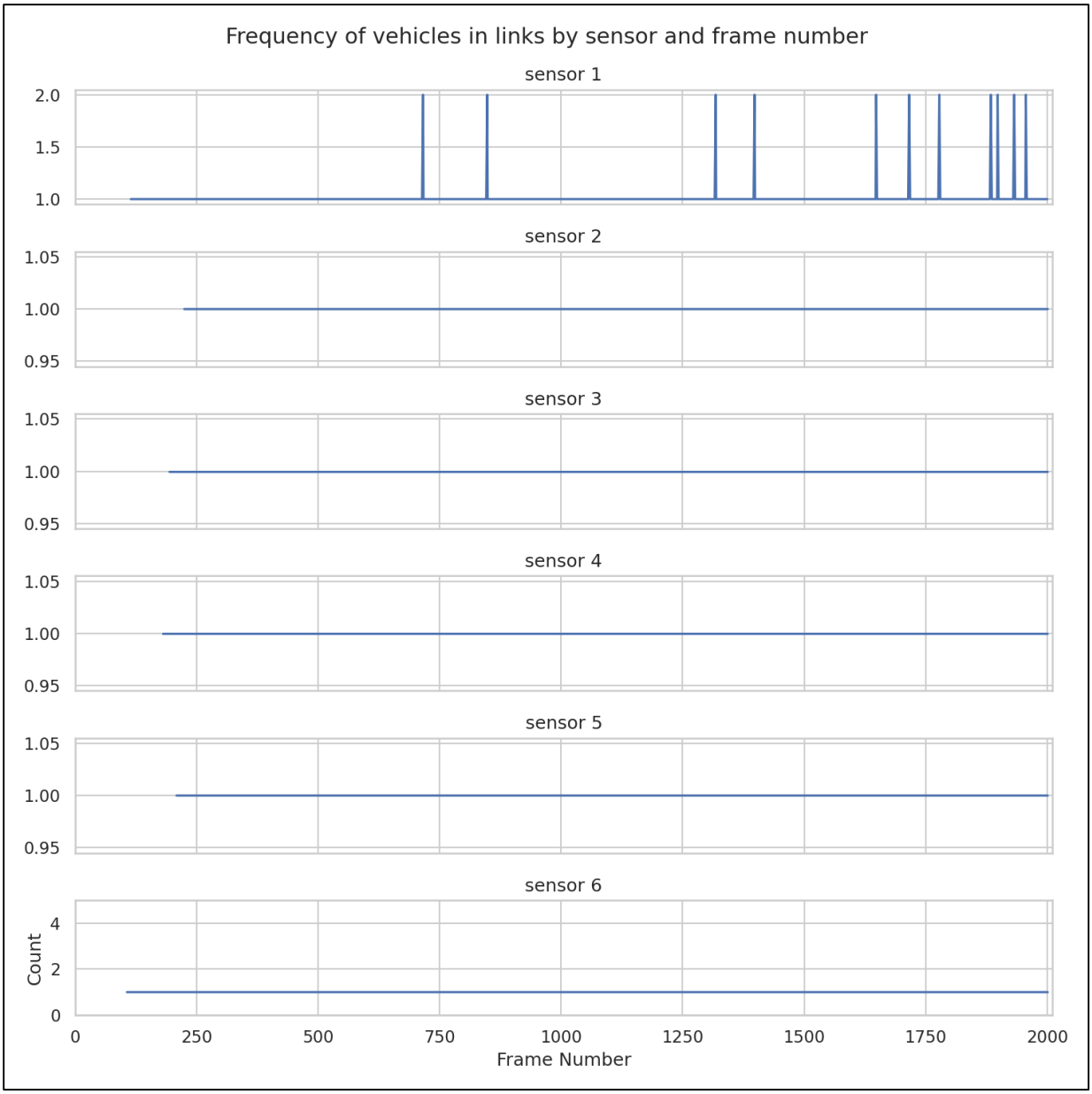

Frequency of vehicles in links by sensor and frame number
sensor 1
2.0
1.5
1.0
sensor 2
1.05
1.00
0.95
sensor 3
1.05
1.00
0.95
sensor 4
1.05
1.00
0.95
sensor 5
1.05
1.00
0.95
sensor 6
Count
4
2
0
0
250
500
750
1000
1250
1500
1750
2000
Frame Number

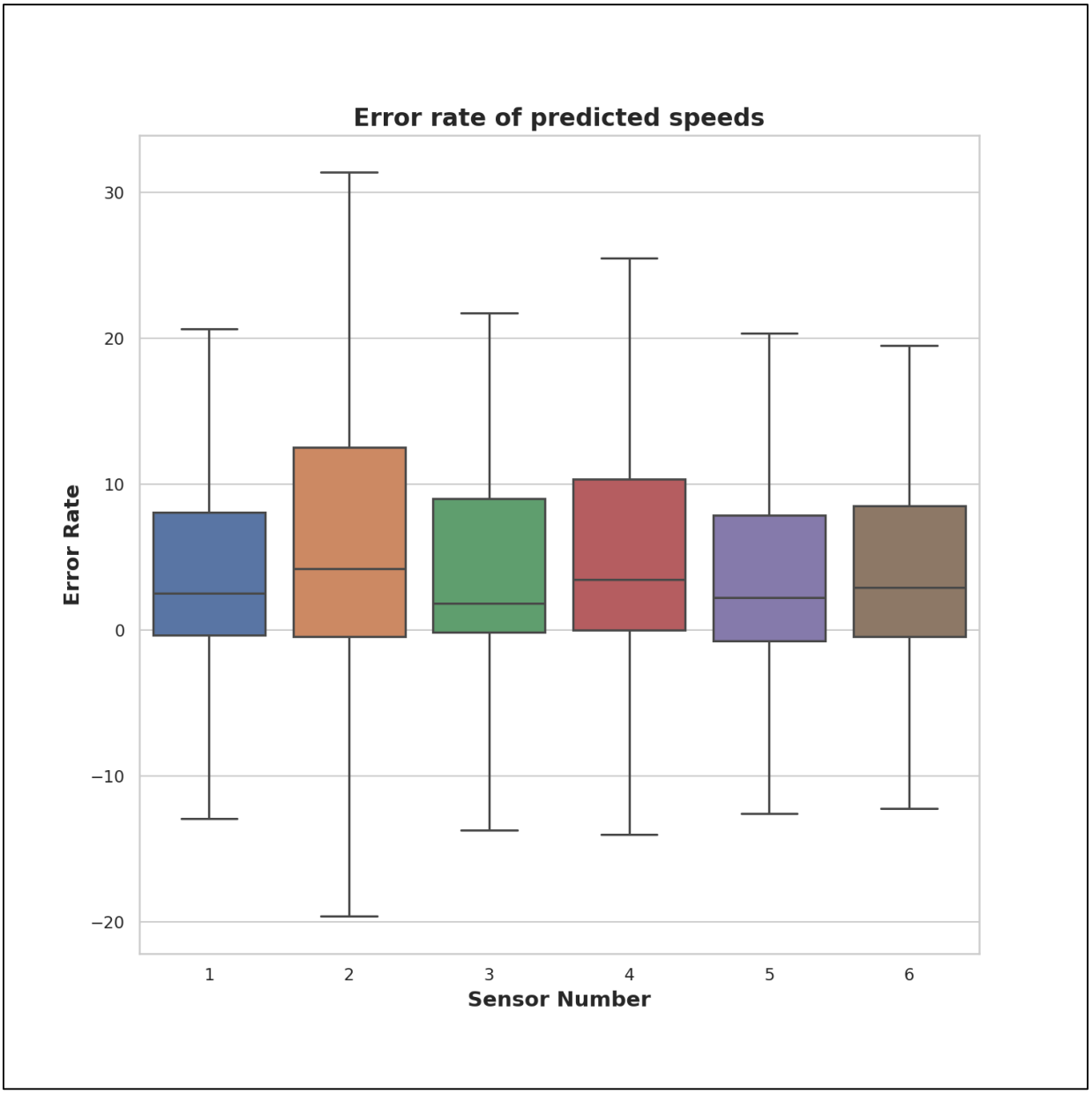

Error rate of predicted speeds
Error Rate
30
20
10
0
−10
−20
−30
1
2
3
4
5
6
Sensor Number


Error rate of predicted speeds per frame
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate
frame_number
carla_id
529
531
534
535
536
537
538
539
542
545
546
548
549
550
554
555
556
564
565
566

D-1000

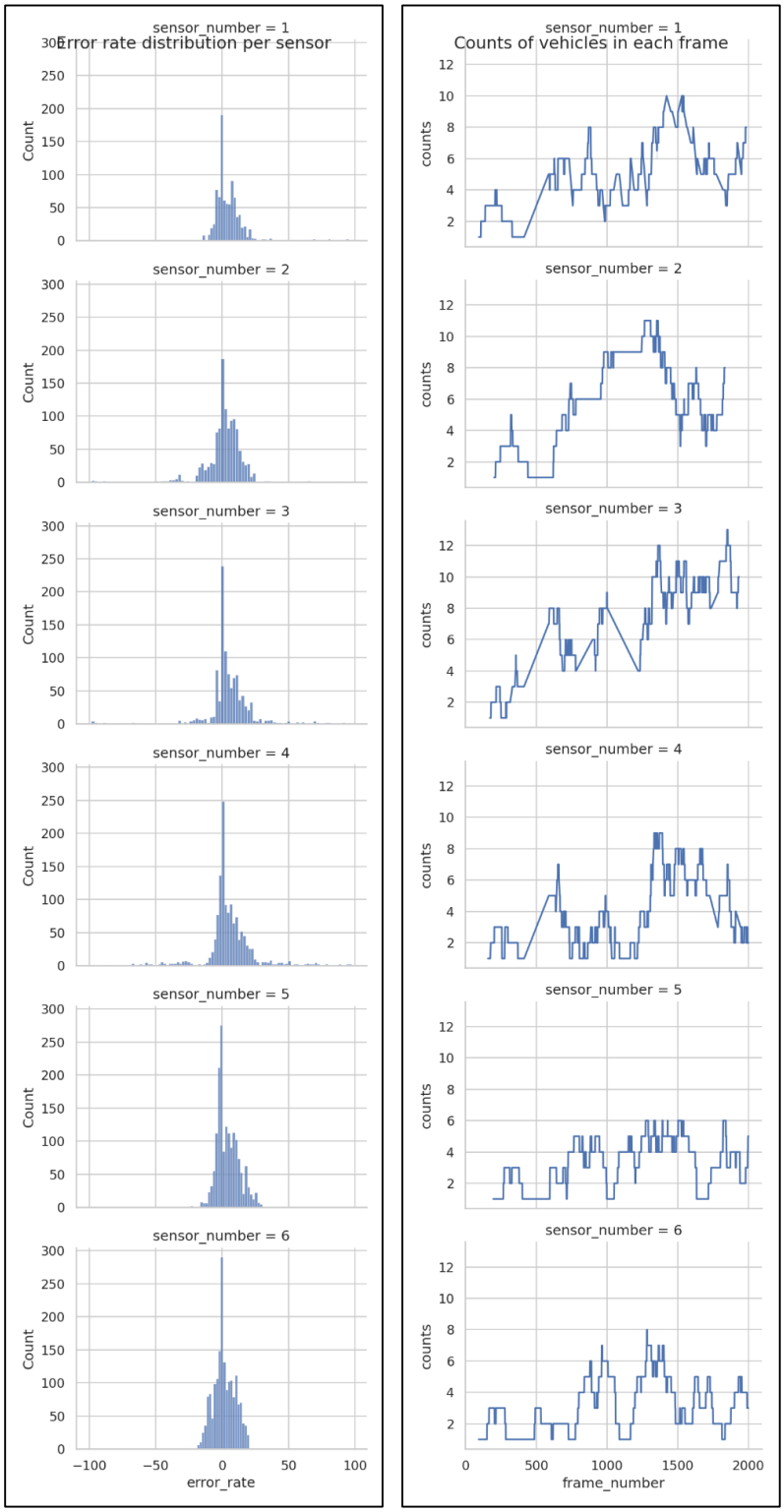

sensor_number = 1
Error rate distribution per sensor
Count
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate
sensor_number = 1
Counts of vehicles in each frame
counts
frame_number

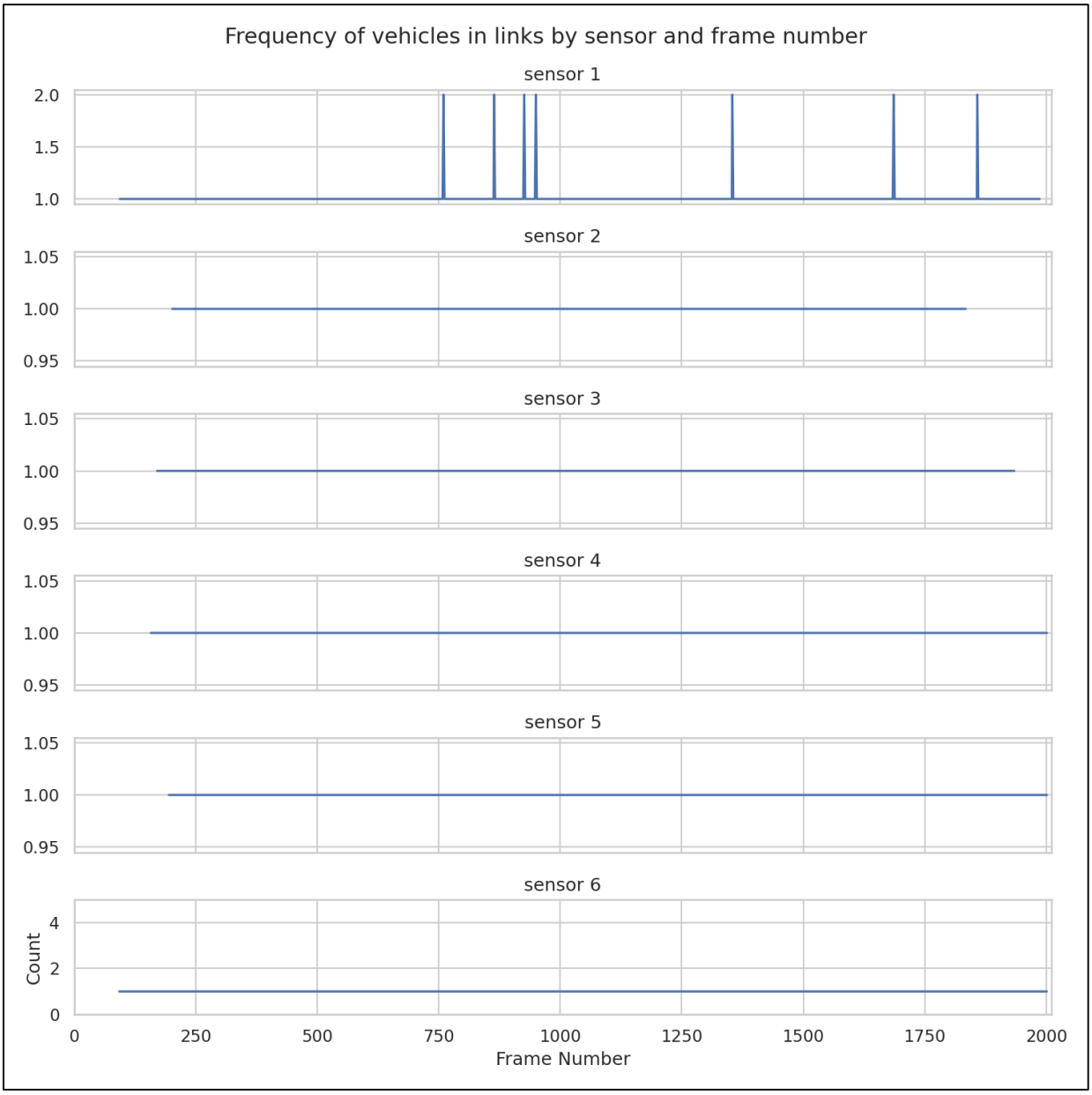

Frequency of vehicles in links by sensor and frame number
sensor 1
2.0
1.5
1.0
sensor 2
1.05
1.00
0.95
sensor 3
1.05
1.00
0.95
sensor 4
1.05
1.00
0.95
sensor 5
1.05
1.00
0.95
sensor 6
Count
4
2
0
0
250
500
750
1000
1250
1500
1750
2000
Frame Number

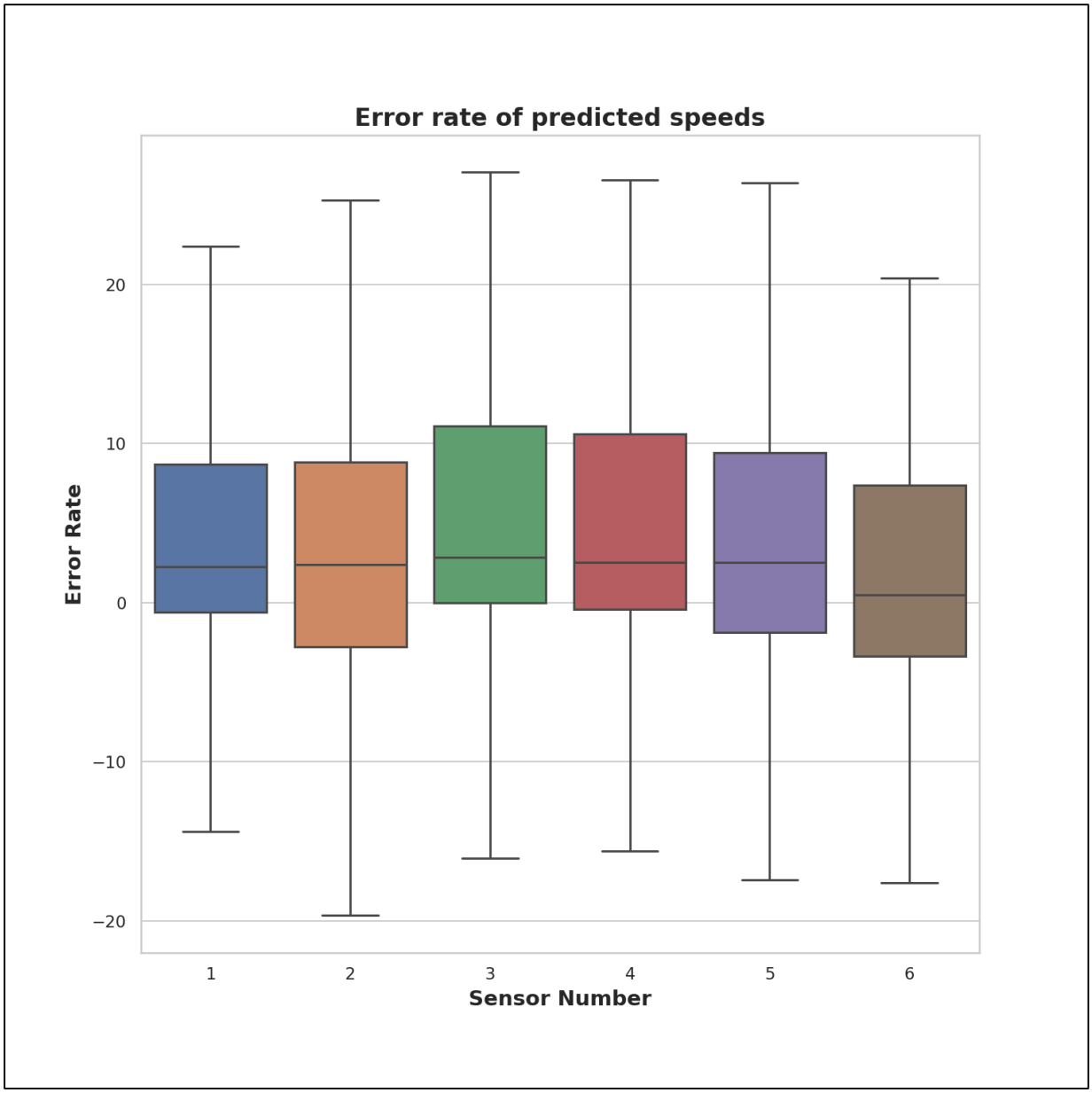

Error rate of predicted speeds
Error Rate
30
20
10
0
−10
−20
1
2
3
4
5
6
Sensor Number


Error rate of predicted speeds per frame
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate
frame_number
carla_id
703
705
707
708
709
710
711
712
716
719
720
722
723
724
727
728
730
745
748
750
751
762

D-1250

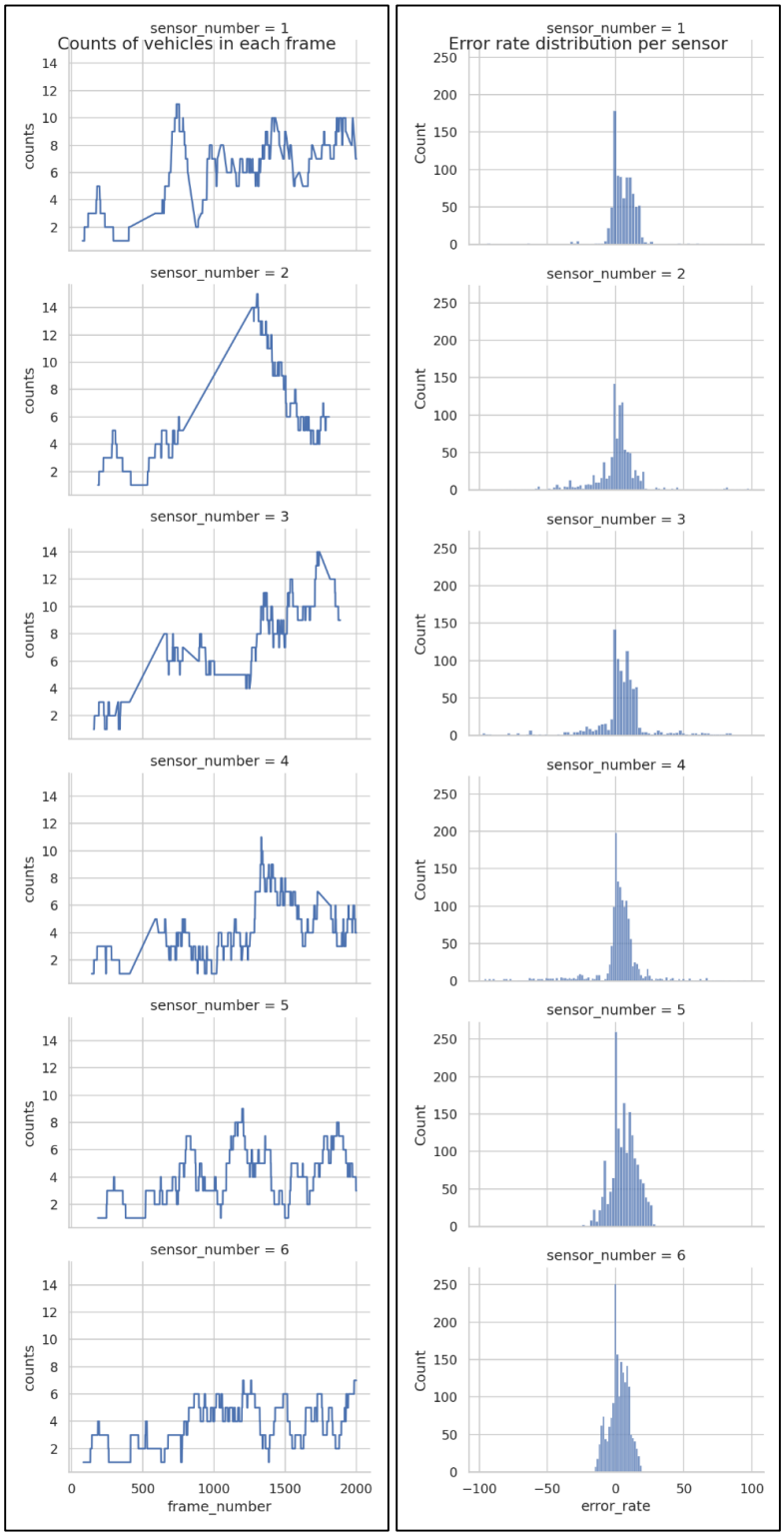

Error rate distribution per sensor
Counts of vehicles in each frame
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
Count
counts
error_rate
frame_number

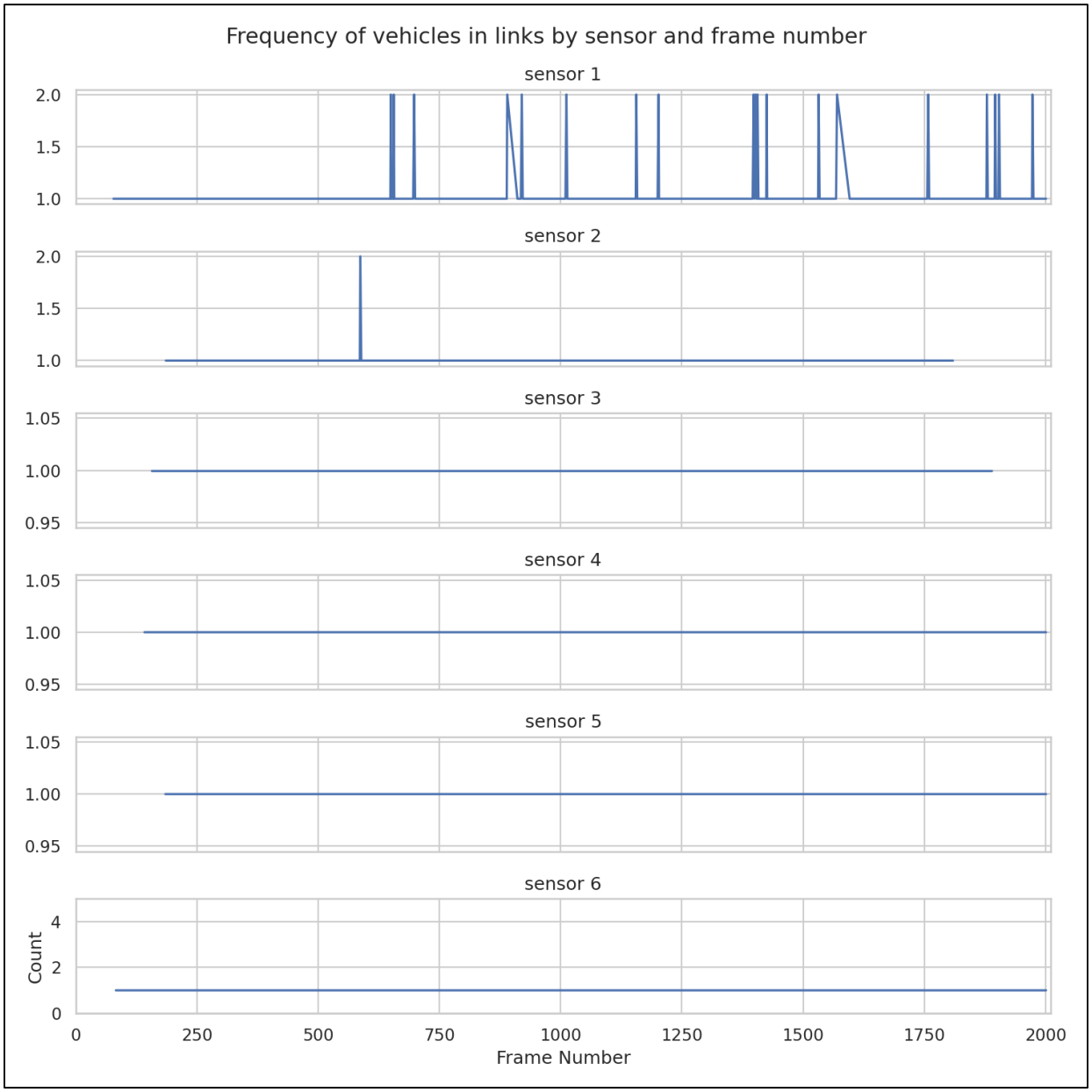

Frequency of vehicles in links by sensor and frame number
sensor 1
2.0
1.5
1.0
sensor 2
1.05
1.00
0.95
sensor 3
1.05
1.00
0.95
sensor 4
1.05
1.00
0.95
sensor 5
1.05
1.00
0.95
sensor 6
Count
4
2
0
0
250
500
750
1000
1250
1500
1750
2000
Frame Number

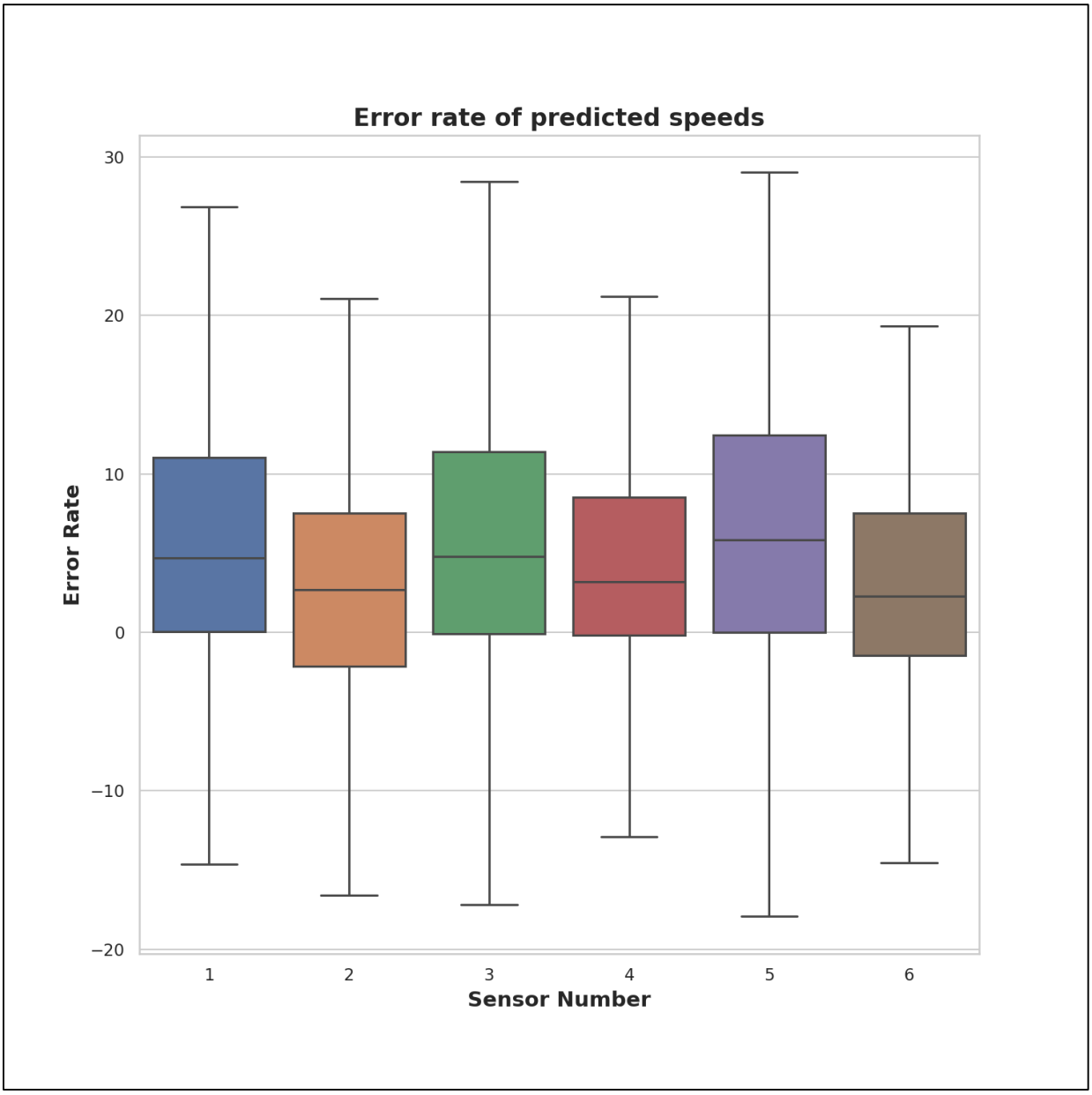

Error rate of predicted speeds
Error Rate
20
10
0
−10
−20
1
2
3
4
5
6
Sensor Number

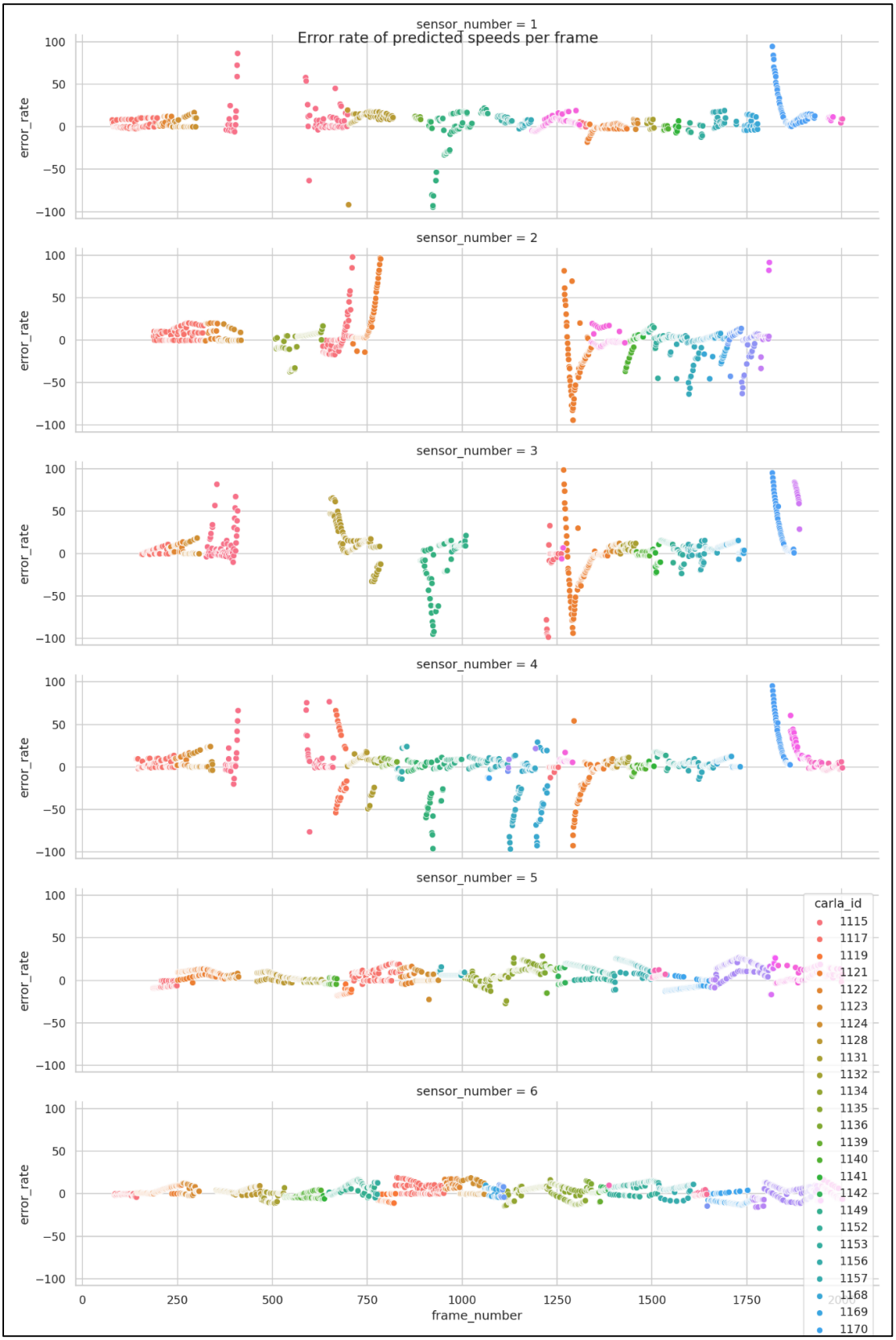

Error rate of predicted speeds per frame
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate
frame_number
carla_id
903
905
907
908
909
910
911
912
916
919
920
922
923
924
927
928
929
930
938
940
943
945
956

D-1500

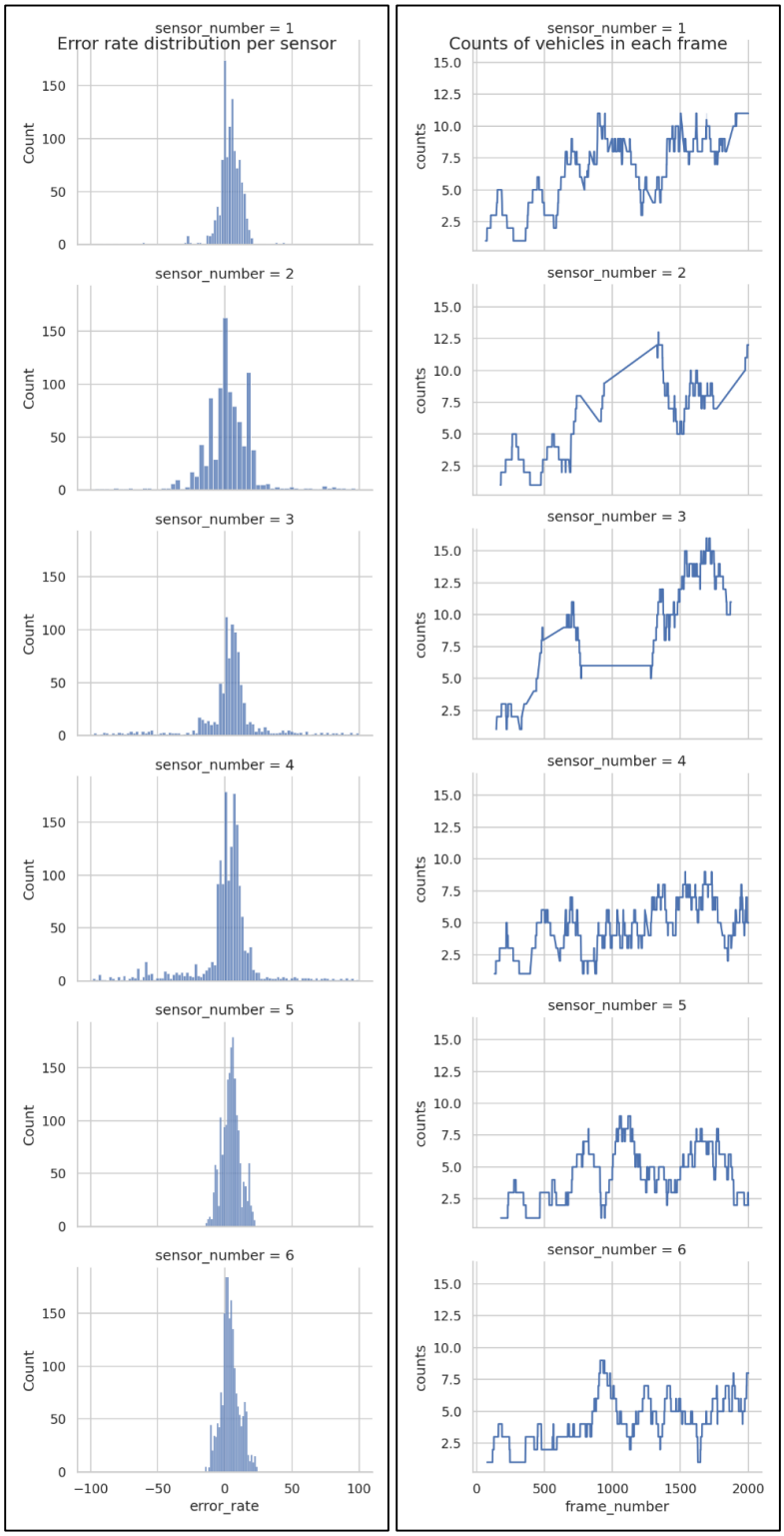

sensor_number = 1
Counts of vehicles in each frame
counts
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
frame_number
sensor_number = 1
Error rate distribution per sensor
Count
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate

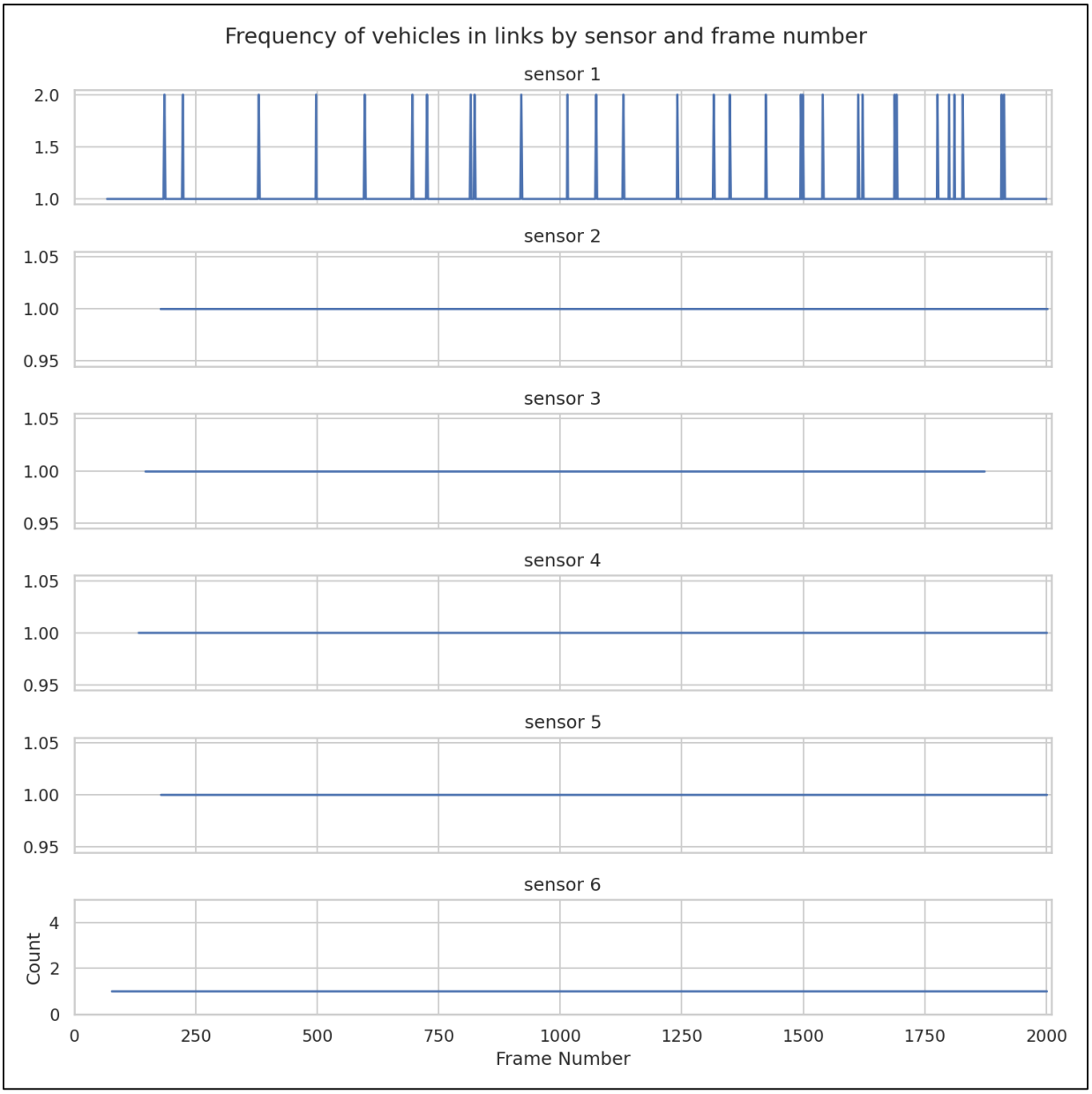

Frequency of vehicles in links by sensor and frame number
sensor 1
2.0
1.5
1.0
sensor 2
2.0
1.5
1.0
sensor 3
1.05
1.00
0.95
sensor 4
1.05
1.00
0.95
sensor 5
1.05
1.00
0.95
sensor 6
Count
4
2
0
0
250
500
750
1000
1250
1500
1750
2000
Frame Number

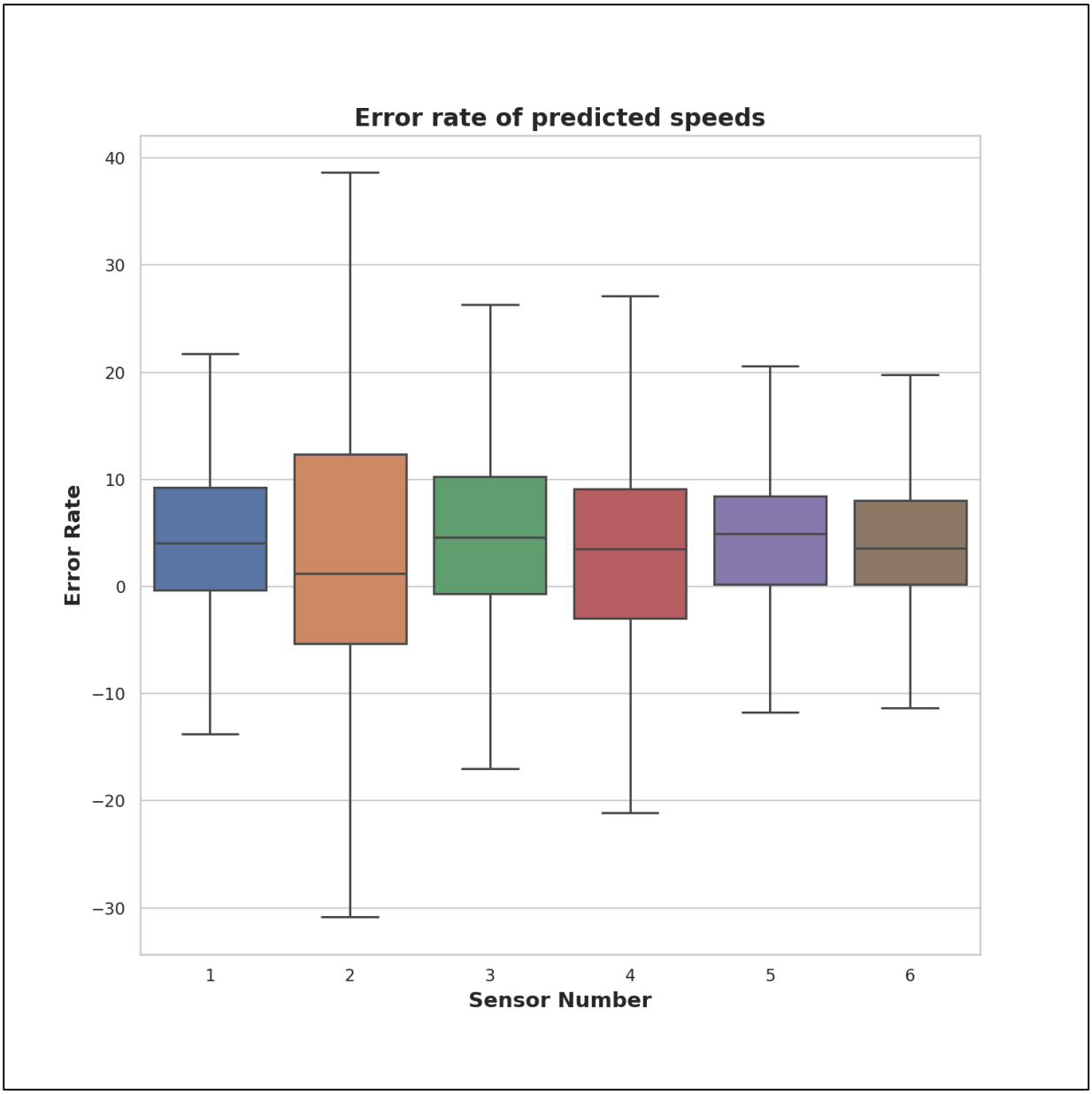

Error rate of predicted speeds
Error Rate
30
20
10
0
−10
−20
1
2
3
4
5
6
Sensor Number


Error rate of predicted speeds per frame
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate
frame_number
carla_id
1115
1117
1119
1121
1122
1123
1124
1128
1131
1132
1134
1135
1136
1139
1140
1141
1142
1149
1152
1153
1156
1157
1168
1169
1170

D-1750

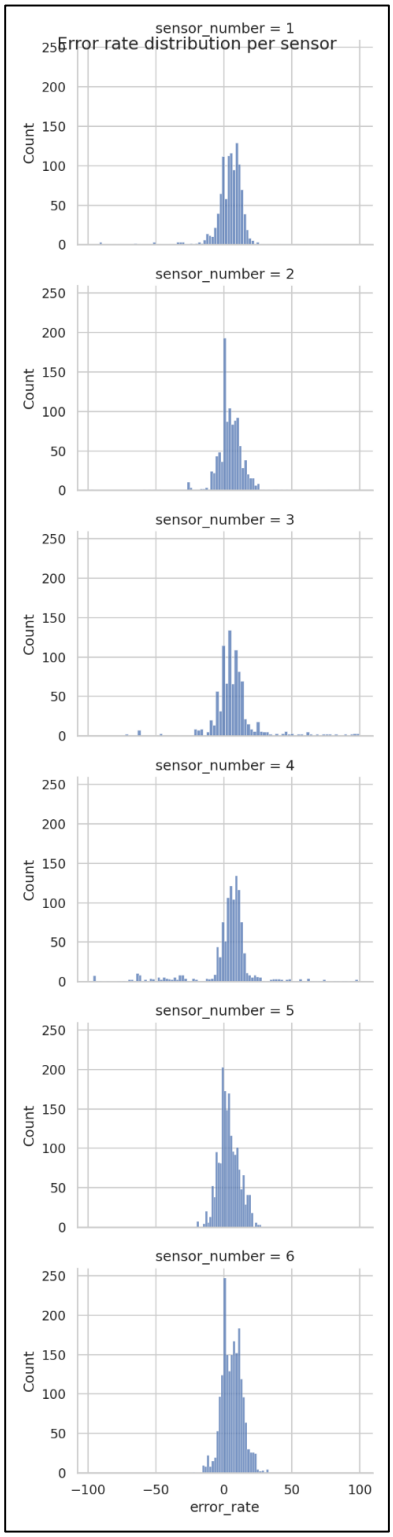

sensor_number = 1
Error rate distribution per sensor
Count
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate
sensor_number = 1
Counts of vehicles in each frame
counts
frame_number

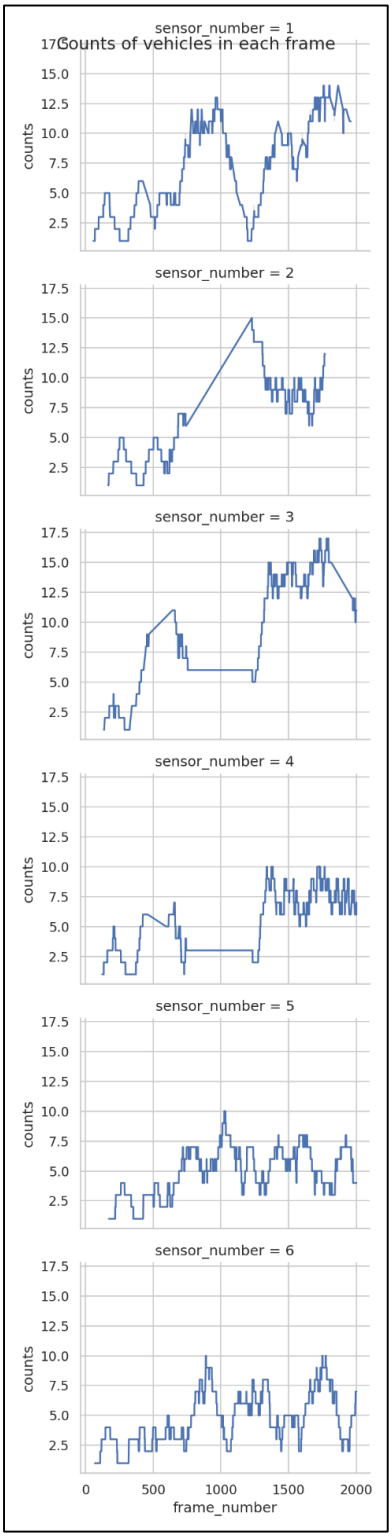

Frequency of vehicles in links by sensor and frame number
sensor 1
2.0
1.5
1.0
sensor 2
1.05
1.00
0.95
sensor 3
1.05
1.00
0.95
sensor 4
1.05
1.00
0.95
sensor 5
1.05
1.00
0.95
sensor 6
Count
4
2
0
0
250
500
750
1000
1250
1500
1750
2000
Frame Number

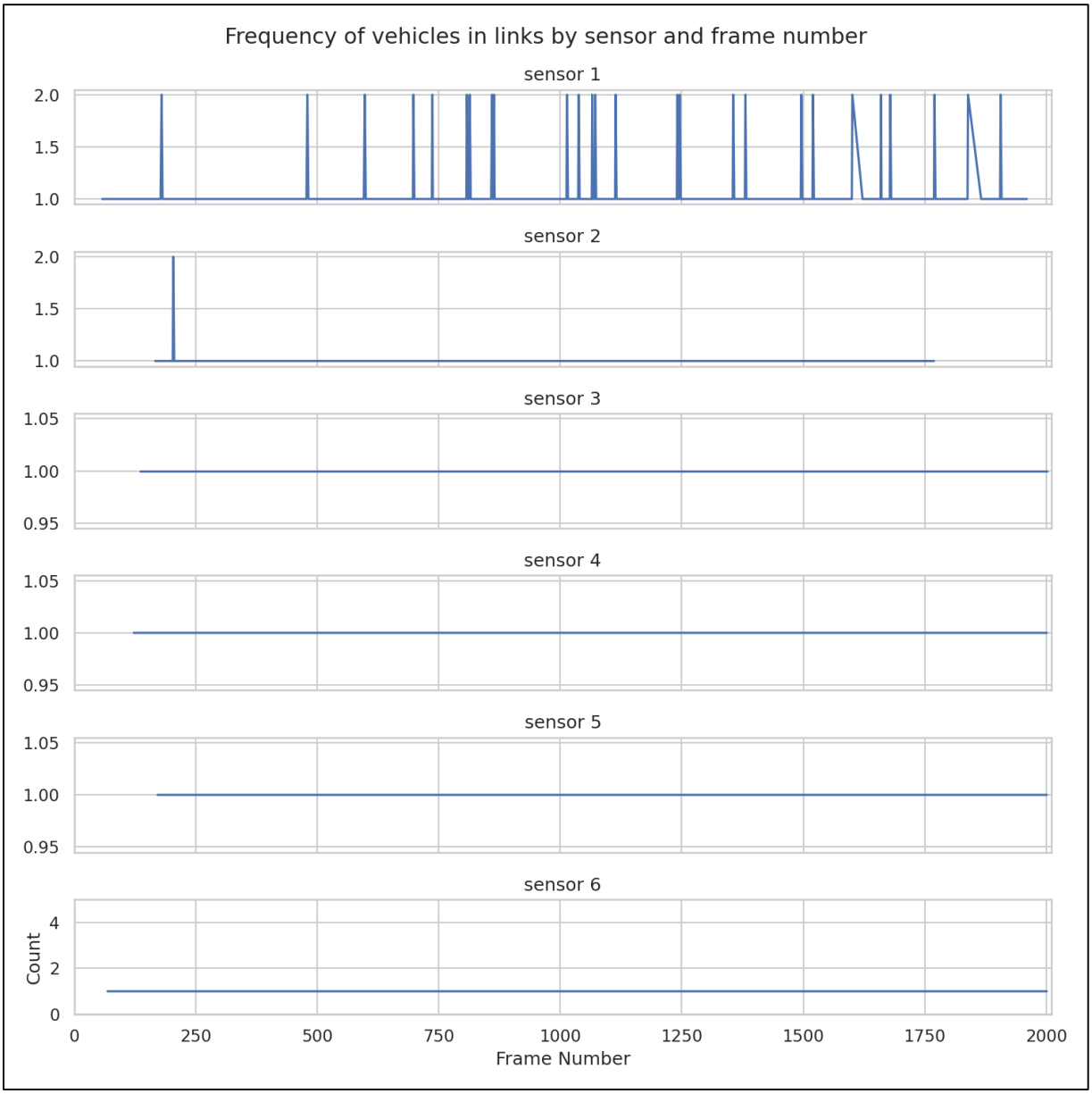

Error rate of predicted speeds
Error Rate
40
30
20
10
0
−10
−20
−30
1
2
3
4
5
6
Sensor Number

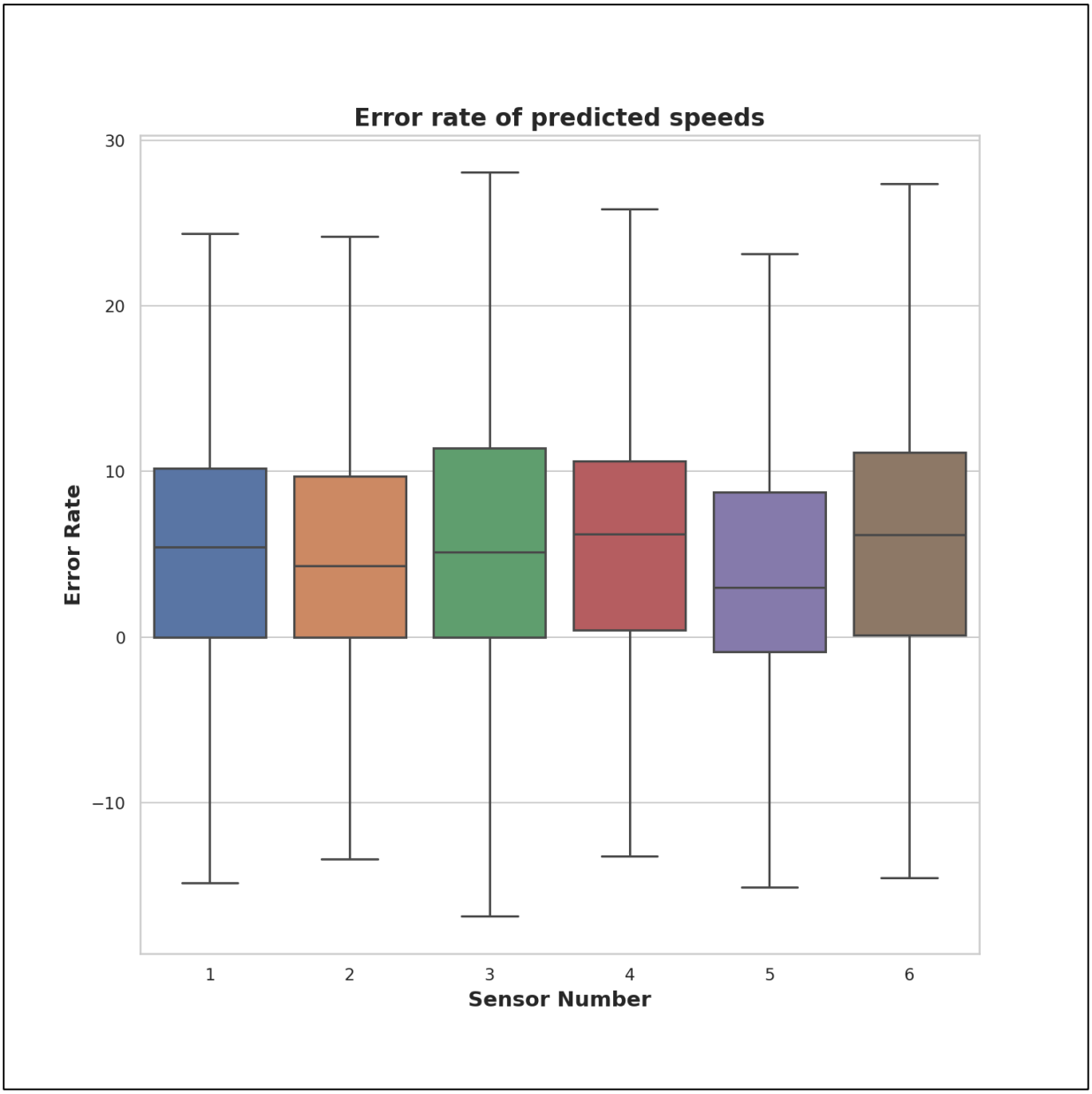

Error rate of predicted speeds per frame
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate
frame_number
carla_id
1347
1349
1352
1353
1355
1356
1360
1363
1364
1366
1371
1373
1380
1385
1388
1391
1395
1401
1402
1405
1407
1409
1410
1411

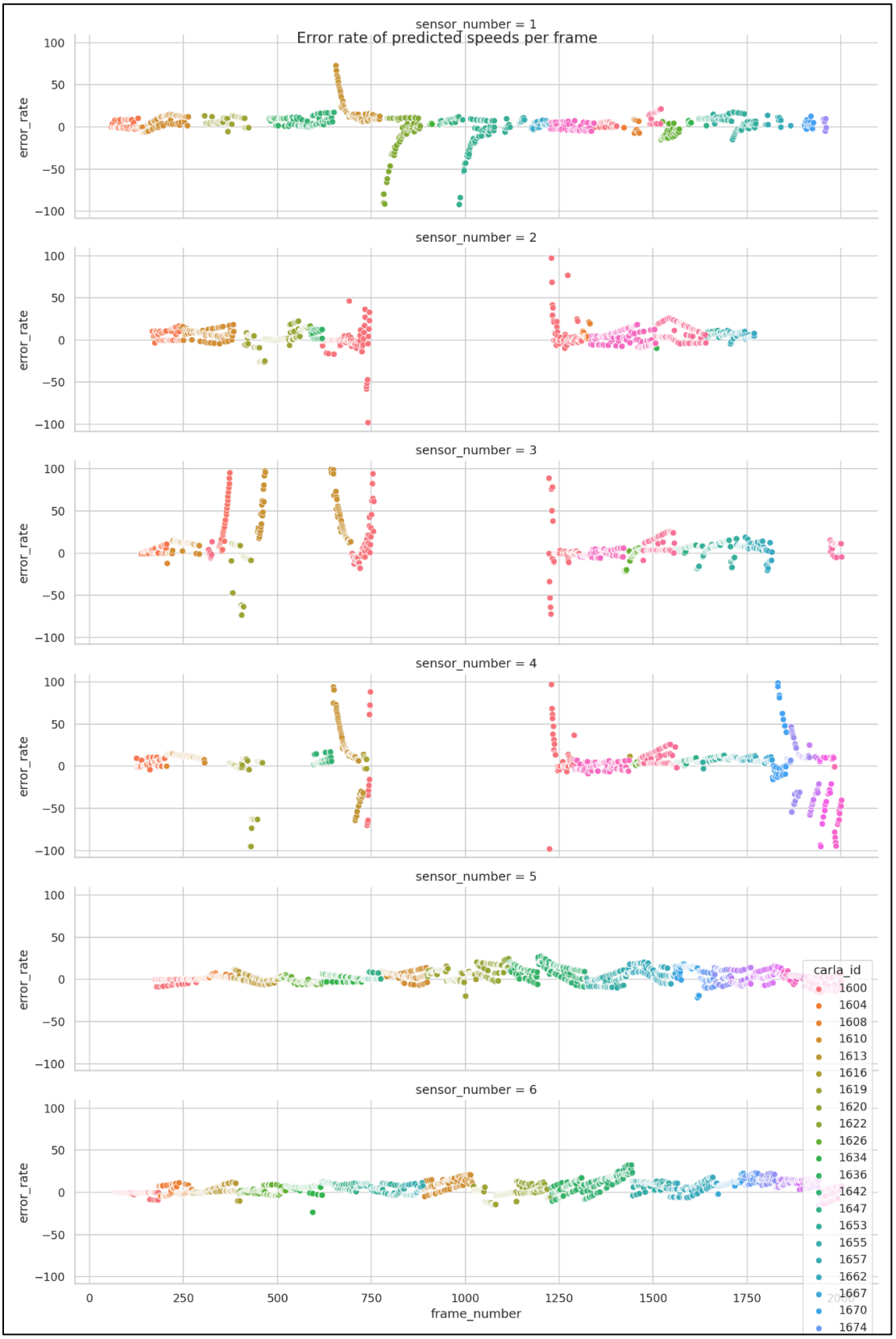
D-2000


Error rate distribution per sensor
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
Count
error_rate

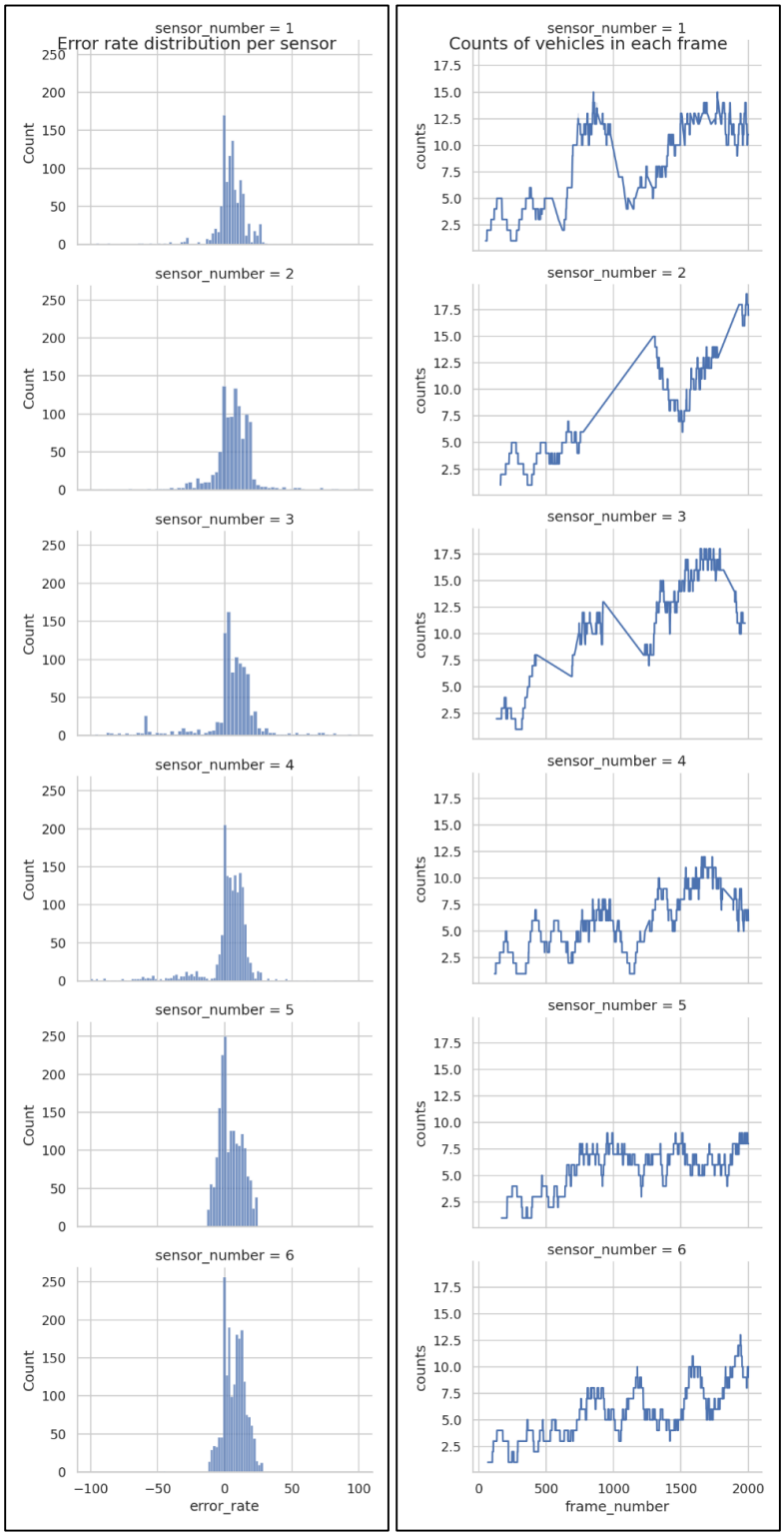

Counts of vehicles in each frame
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
counts
frame_number

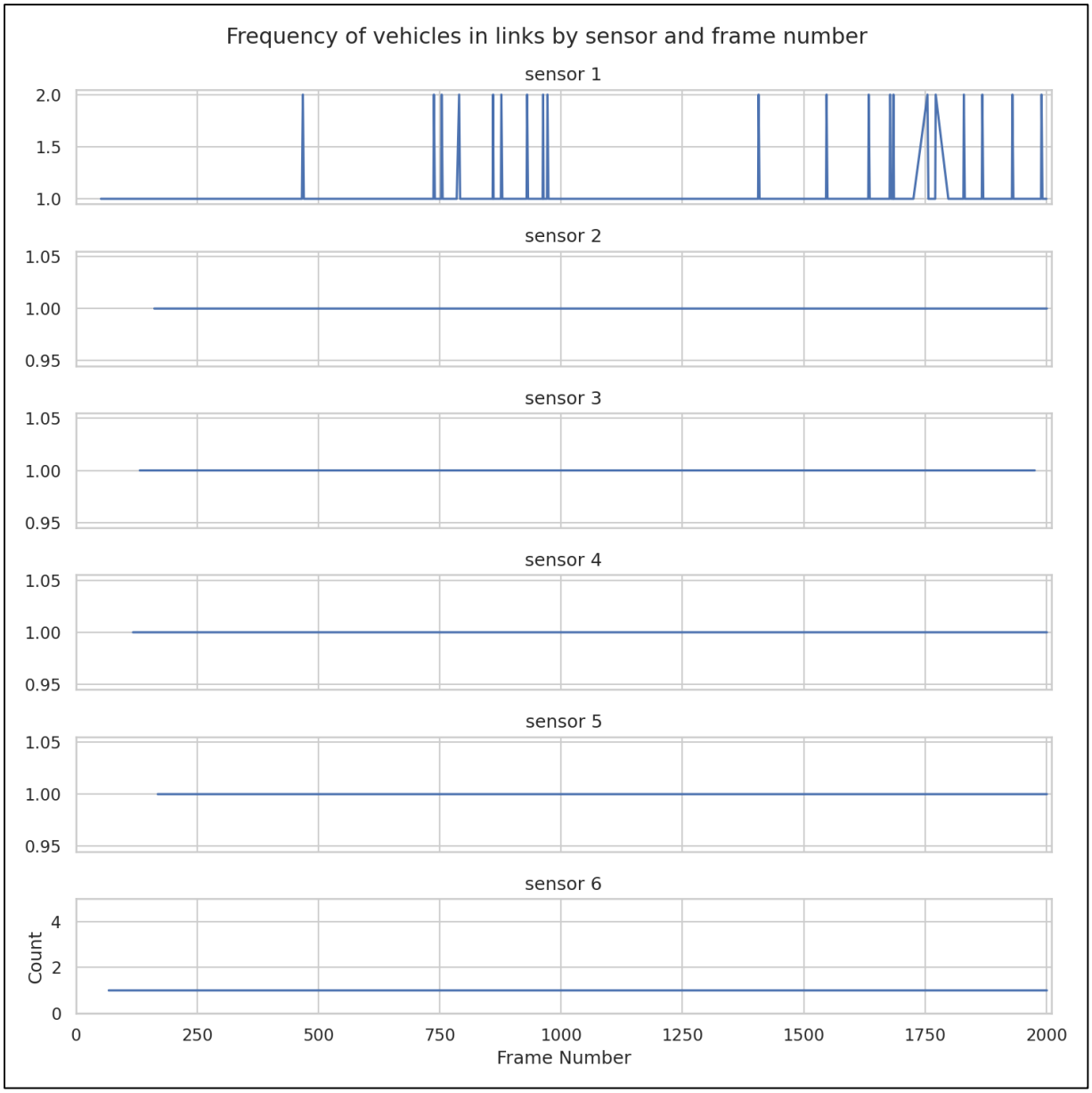

Frequency of vehicles in links by sensor and frame number
sensor 1
2.0
1.5
1.0
sensor 2
2.0
1.5
1.0
sensor 3
1.05
1.00
0.95
sensor 4
1.05
1.00
0.95
sensor 5
1.05
1.00
0.95
sensor 6
Count
4
2
0
0
250
500
750
1000
1250
1500
1750
2000
Frame Number

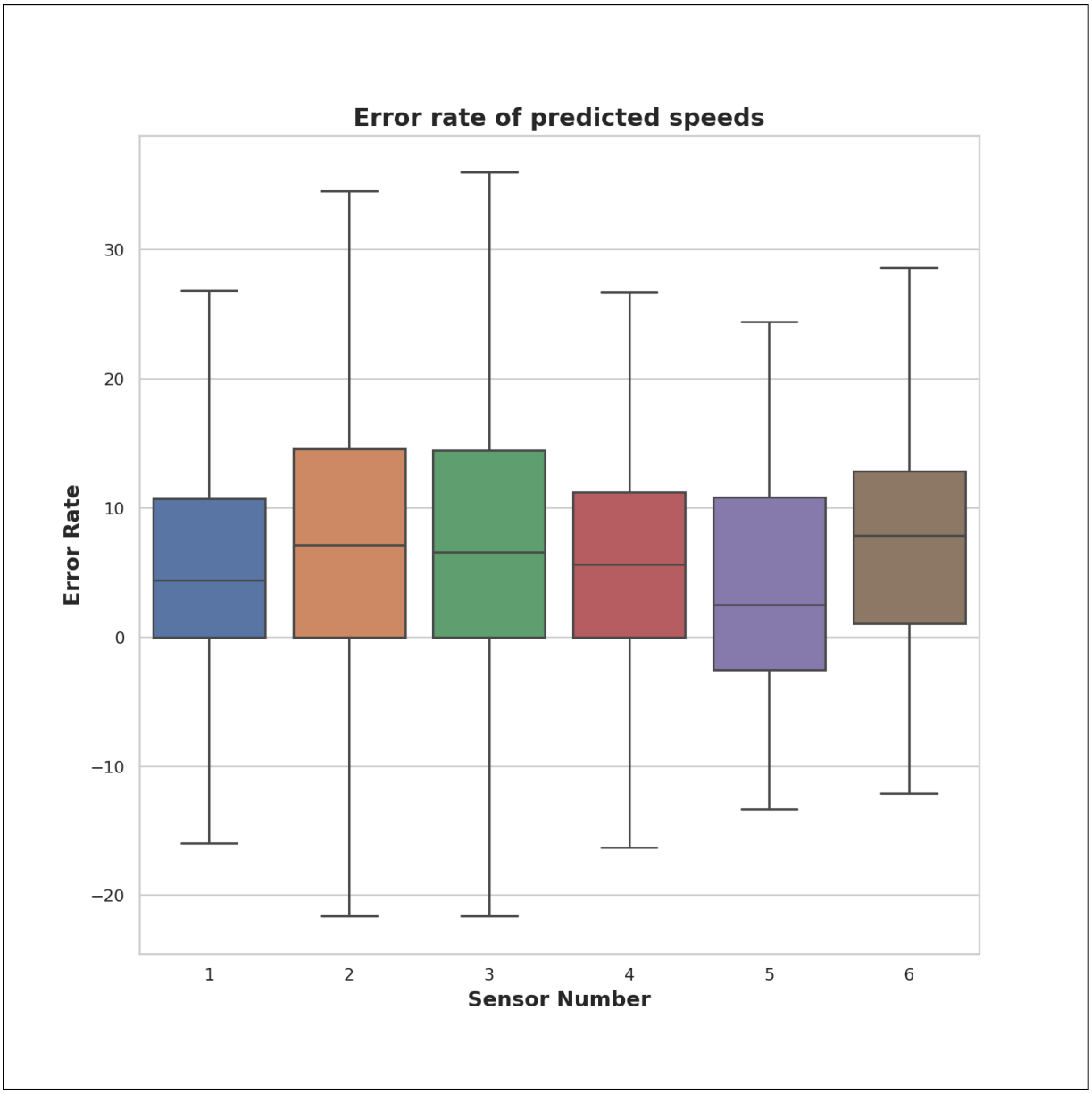

Error rate of predicted speeds
30
20
10
0
−10
Error Rate
1
2
3
4
5
6
Sensor Number


Error rate of predicted speeds per frame
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate
frame_number
carla_id
1600
1604
1608
1610
1613
1616
1619
1620
1622
1626
1634
1636
1642
1647
1653
1655
1657
1662
1667
1670
1674

D-2250

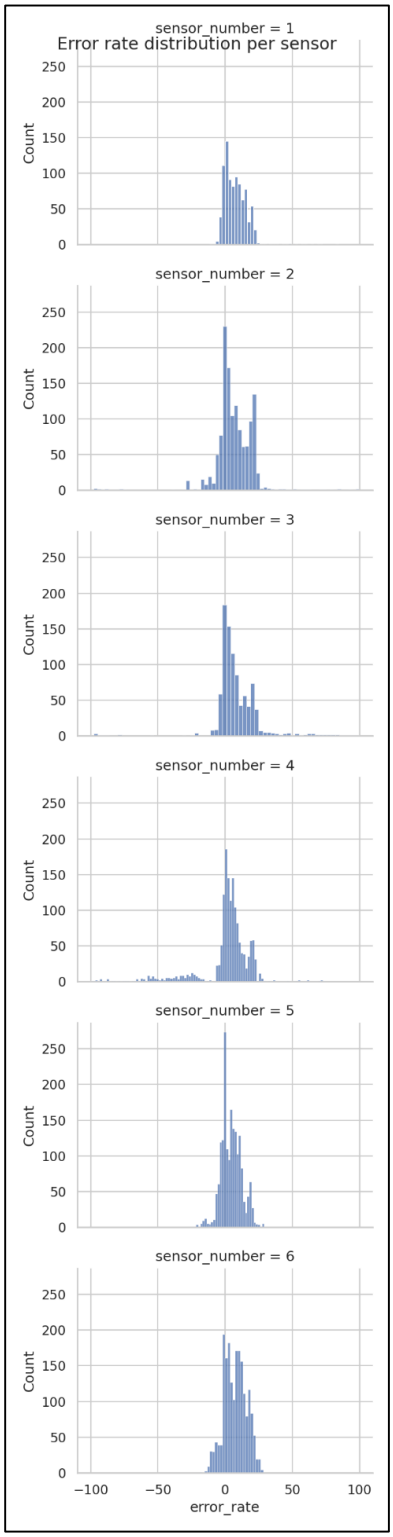

sensor_number = 1
Error rate distribution per sensor
Count
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate
sensor_number = 1
Counts of vehicles in each frame
counts
frame_number

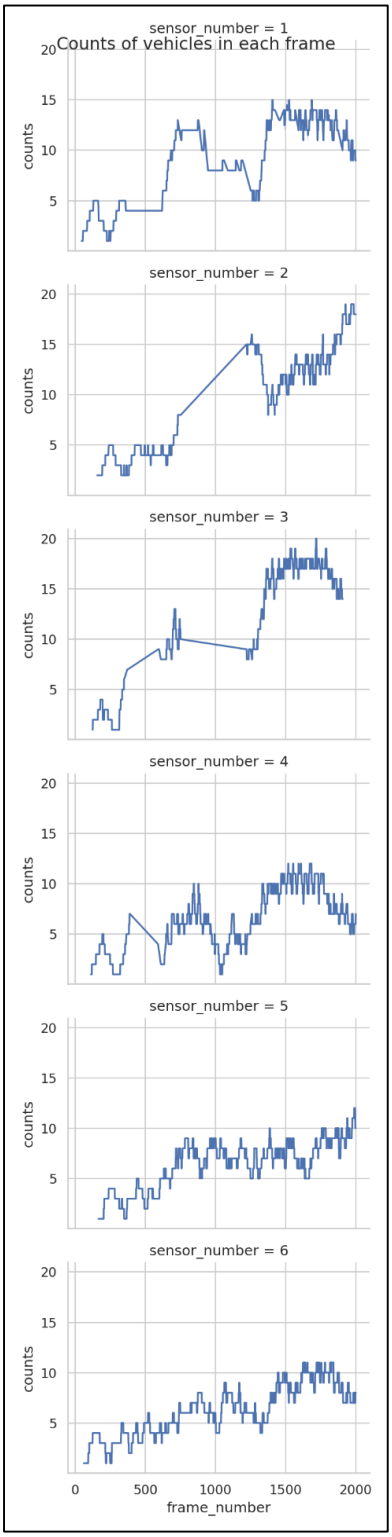

Frequency of vehicles in links by sensor and frame number
sensor 1
2.0
1.5
1.0
sensor 2
1.05
1.00
0.95
sensor 3
1.05
1.00
0.95
sensor 4
1.05
1.00
0.95
sensor 5
1.05
1.00
0.95
sensor 6
Count
4
2
0
0
250
500
750
1000
1250
1500
1750
2000
Frame Number

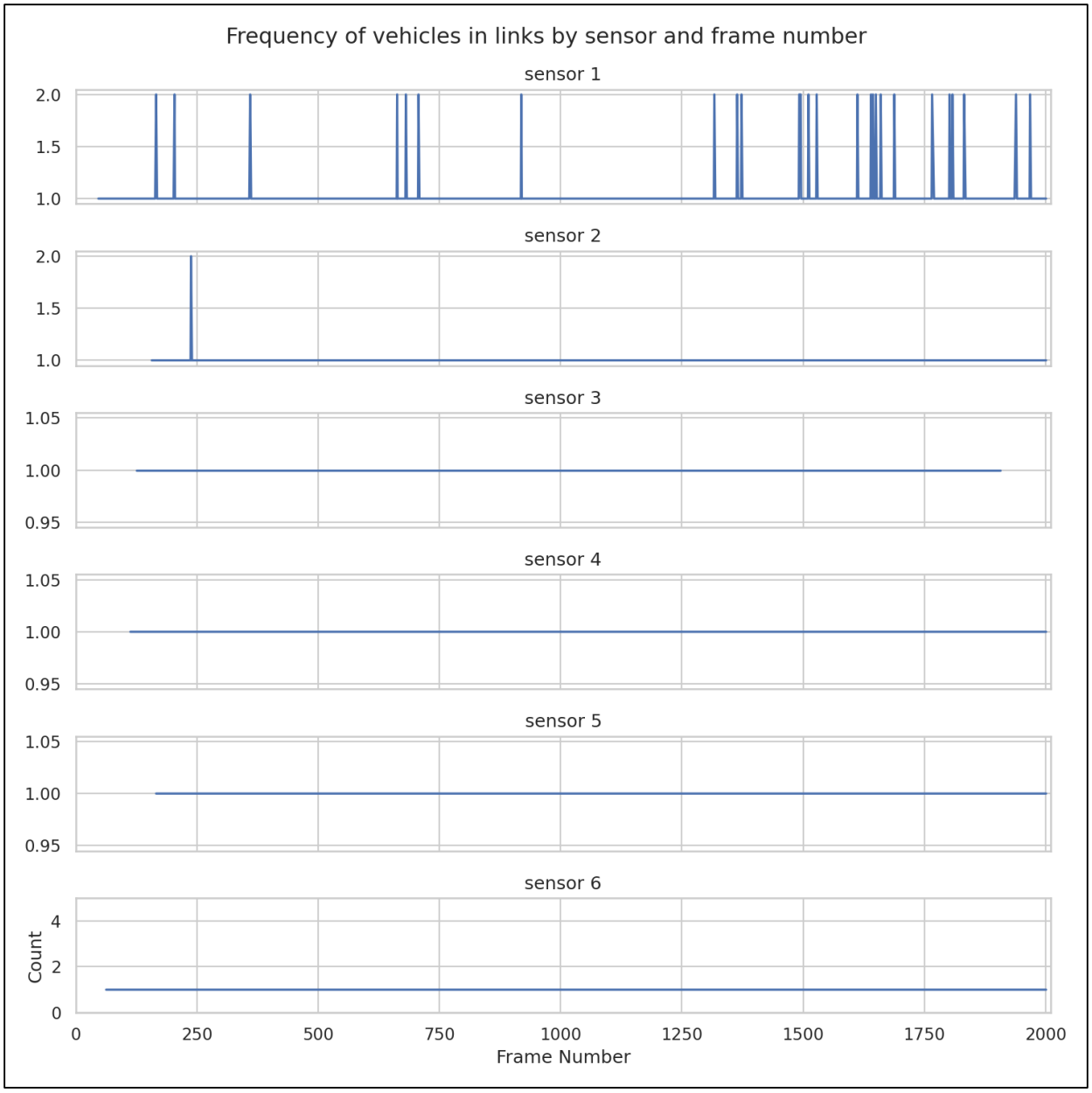

Error rate of predicted speeds
Error Rate
30
20
10
0
−10
−20
1
2
3
4
5
6
Sensor Number

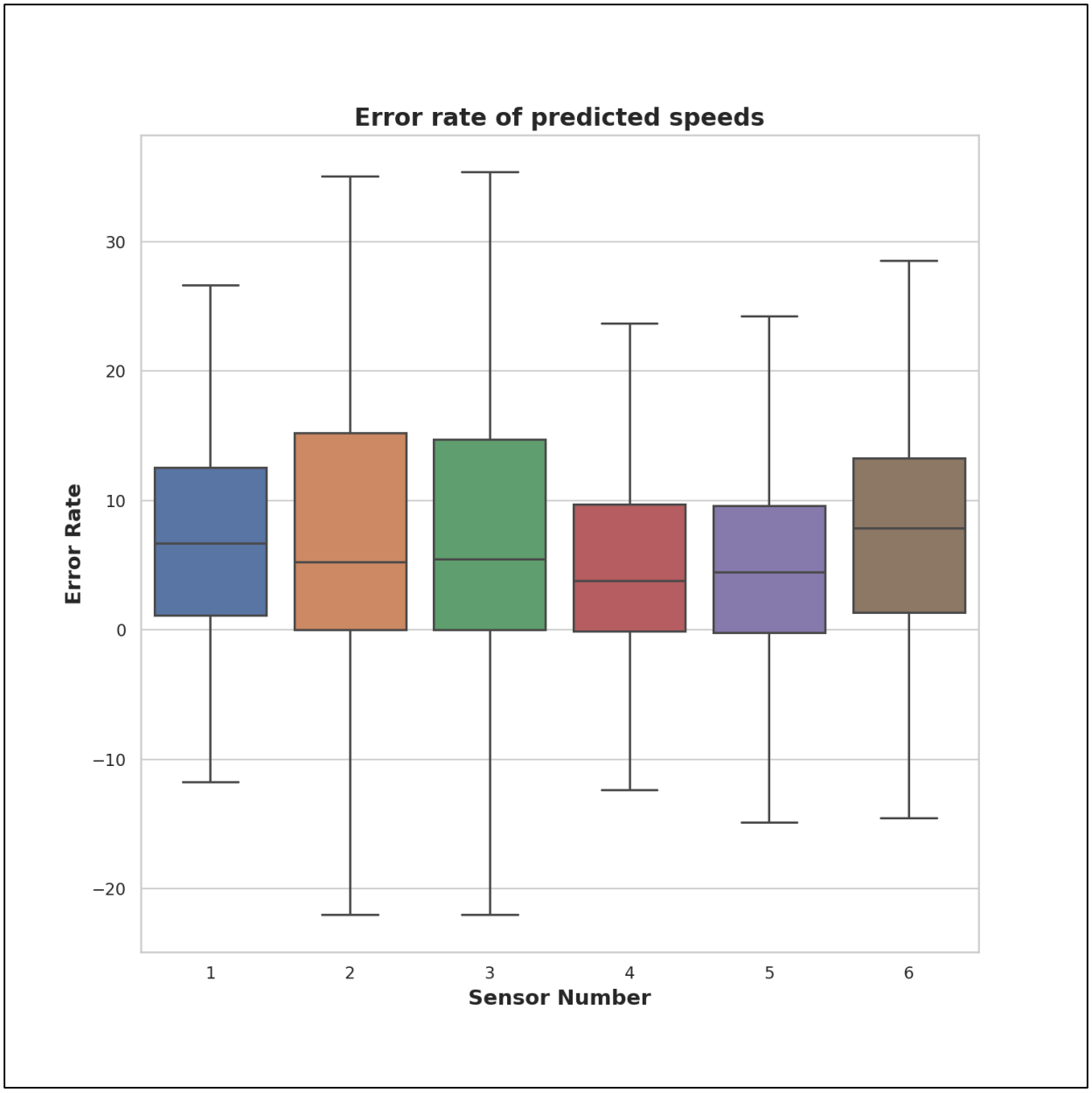

Error rate of predicted speeds per frame
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate
frame_number
carla_id
1882
1884
1885
1888
1889
1890
1891
1893
1895
1898
1899
1901
1902
1908
1909
1916
1917
1920
1922
1924
1927
1929
1930
1935
1937

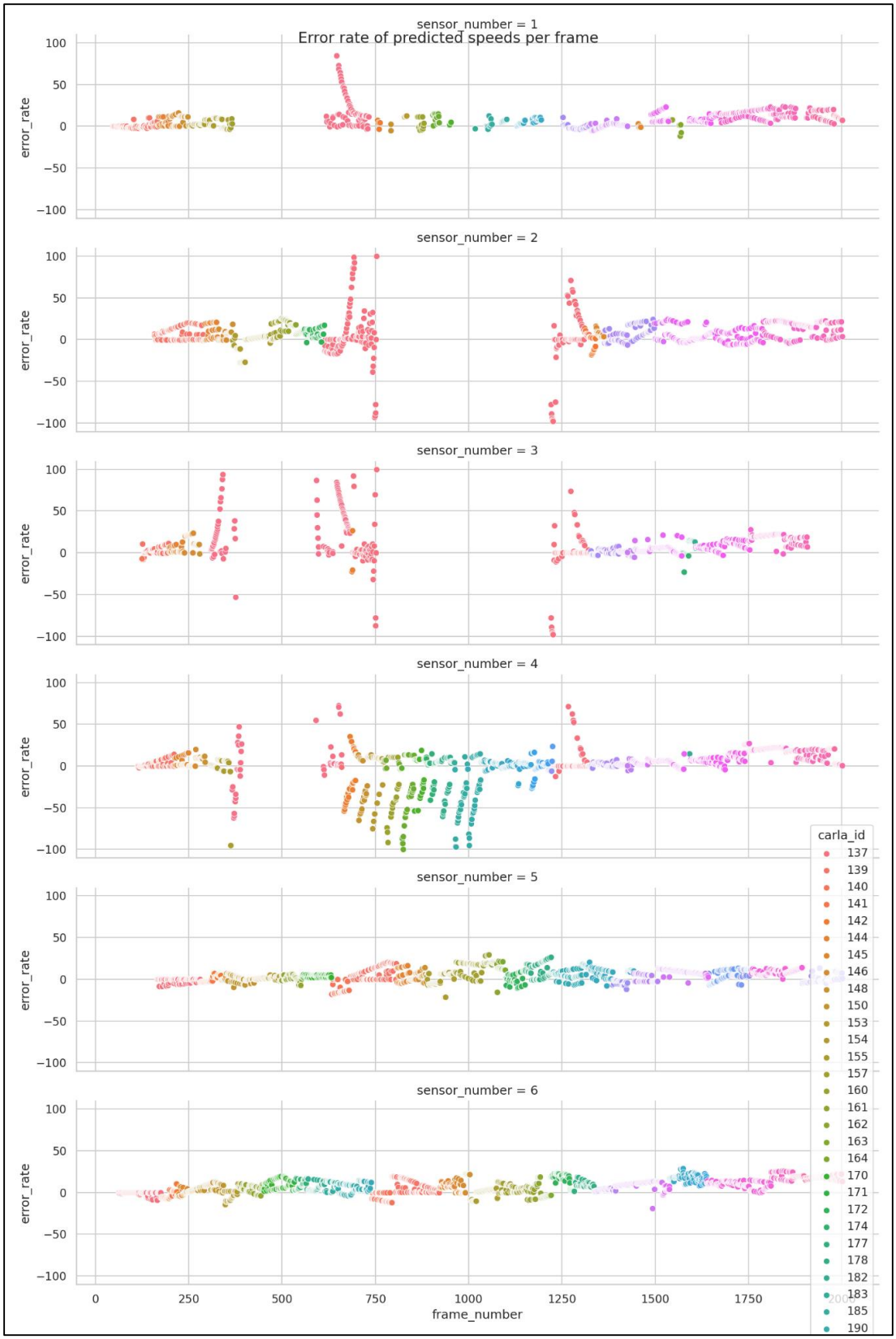
D-2500


Error rate distribution per sensor
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
Count
error_rate


Counts of vehicles in each frame
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
counts
frame_number


Frequency of vehicles in links by sensor and frame number
sensor 1
2.0
1.5
1.0
sensor 2
2.0
1.5
1.0
sensor 3
1.05
1.00
0.95
sensor 4
1.05
1.00
0.95
sensor 5
1.05
1.00
0.95
sensor 6
Count
4
2
0
0
250
500
750
1000
1250
1500
1750
2000
Frame Number


Error rate of predicted speeds
Error Rate
30
20
10
0
−10
−20
1
2
3
4
5
6
Sensor Number


Error rate of predicted speeds per frame
sensor_number = 1
sensor_number = 2
sensor_number = 3
sensor_number = 4
sensor_number = 5
sensor_number = 6
error_rate
frame_number
carla_id
137
139
140
141
142
144
145
146
148
150
153
154
155
157
160
161
162
163
164
170
171
172
174
177
178
182
183
185
190