Informatics

Simulation-Based Disaggregation of Train Delay Data Using Graph Neural Networks

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

066 645 Data Science

by

Maximilian Viehauser, BSc
Registration Number 11945353

to the Faculty of Informatics

at the TU Wien

Advisor: Dr.techn. Nikolas Popper
Assistance: Dr.techn. Martin Bicher
Dipl.Ing. Matthias Rößler

Vienna, September 2, 2024

Maximilian Viehauser Nikolas Popper

Erklärung zur Verfassung der Arbeit

Maximilian Viehauser, BSc

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe
Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel" habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.

Wien, 2. September 2024

Maximilian Viehauser

Acknowledgements

I would like to thank my supervisors, Martin Bicher, Nikolas Popper, and Matthias Rößler, for their guidance during the work on this thesis. A special thanks goes to Martin Bicher, who gradually introduced me to the world of Modeling and Simulation through two university courses that sparked a great interest in me, then allowed me to tutor the courses myself, and finally guided me in writing a thesis in this field. Further gratitude goes to Matthias Rößler, a specialist in the agent-based model of the Austrian Railway system, who provided me with a thorough introduction to this sophisticated model. He was always willing to answer my questions, helped me understand the mechanisms of the model, and successfully guided my research.

Kurzfassung

Diese Arbeit untersucht, wie sich aggregierte Zugverspätungen (AGD) mithilfe simulationsbasierter Ansätze und Graph Neural Networks (GNN) in primäre (PD) und sekundäre (SD) Verspätungsanteile zerlegen lassen. Dabei wird untersucht, ob Gated Graph Convolutional Networks (GatedGCN), eine spezielle Form von GNN, geeignet sind, um AGD zuverlässig zu disaggregieren.

Für die Reduzierung von Zugverspätungen ist es von Bedeutung, zwischen PD und SD zu unterscheiden, da sich diese Verspätungstypen durch unterschiedliche Interventionen minimieren lassen. PD sind Verspätungen, die durch externe Faktoren wie Wetter oder Infrastrukturprobleme verursacht werden, während SD durch Interaktionen innerhalb des Bahnnetzes entstehen. Obwohl ein Großteil der Zugbetreiber Verspätungen dokumentieren, wird jedoch nicht erfasst, ob es sich um PD oder SD handelt. Um verspätungsrobuste Fahrpläne zu erstellen, ist es wichtig zu wissen, in welchem Umfang eine Verspätung auf PD oder SD zurückzuführen ist. Dementsprechend würde ein Algorithmus, der es ermöglicht, AGD zuverlässig zu disaggregieren, wesentlich zur Optimierung des Verspätungsmanagements beitragen und die Grundlage für präzisere und effektivere Fahrplananpassungen schaffen.

Die Kernfragen der Forschung drehen sich darum, ob ein datenbasierter Ansatz mit GNN zur Aufschlüsselung von AGD geeignet ist. Da keine historischen Daten mit einer verlässlichen Grund Truth verfügbar waren, wurden für die Untersuchung künstlich erzeugte Daten verwendet. Diese wurden mit einem agentenbasierten Simulationsmodell des österreichischen Eisenbahnnetzes generiert. Ein Schwerpunkt der Arbeit lag auf der sorgfältigen Kalibrierung dieses Modells, die es ermöglicht hat, realistische Verspätungsdaten nach der Eingabe eines Zugfahrplans und der primären Verspätungen zu erzeugen.

Die Resultate der Experimente zeigen, dass das GNN die AGD auf den synthetischen Daten in PD und SD effizient und zuverlässig aufteilen kann. In weiteren Experimenten, bei denen das trainierte Modell auf echte Daten angewandt wurde, zeigte das GNN eine geringere Performance als ein naives Vergleichsmodell. Dies deutet darauf hin, dass das GNN eine Überanpassung an die synthetischen Daten aufweist, wodurch es Schwierigkeiten hat, auf die realen Verspätungsdaten zu generalisieren. Zudem wurde deutlich, dass die generierten Daten zu stark von den tatsächlichen historischen Verspätungen abweichen, um allein als Trainingsgrundlage zu dienen. Die Studie kommt zu dem Schluss, dass GNNs grundsätzlich gut zur Aufschlüsselung von AGD geeignet sind, vorausgesetzt, es

stehen präzise Daten zu PD und SD zur Verfügung. Bei der Verwendung synthetischer Daten ist jedoch Vorsicht geboten, da diese die Gefahr einer Überanpassung bergen.

Abstract

Primary Delays (PD) refer to delays caused by external factors, such as weather conditions or infrastructure problems, while Secondary Delays (SD) are caused by interactions within the railway network. The majority of train operators record when delays occur but do not differentiate between PD and SD. In order to create delay-robust timetables, it is important to know the extent to which a delay is attributable to PD or SD.

Accordingly, an algorithm that makes it possible to disaggregate Aggregated Delays (AGD) reliably would significantly contribute to the optimization of delay management and create the basis for more precise and effective timetable adjustments. The central research questions aim to determine whether a data-driven approach using Gated Graph Convolutional Neural Networks (GatedGCN) is suitable for disaggregating AGD. This was investigated using synthetic data generated by an agent-based simulation model of the Austrian railway network due to the absence of real-world data with ground truth. A significant part of the research involved calibrating this agent-based model. The model is capable of generating synthetic delay data after receiving a train timetable and PD as input values. The experimental results indicate that the GatedGCN can efficiently and reliably disaggregate AGD into PD and SD when applied to synthetic data. However, in subsequent experiments where the trained model was applied to real-world delay data, the GatedGCN performed worse than a naive comparison model.

This suggests that the GatedGCN overfitted the synthetic data, making it difficult to generalize to real delay data. In addition, the results show that the generated data significantly deviates from the historical delay data, making it unsuitable as the sole data source for GatedGCN training. The study suggests that GatedGCNs are well-suited for the disaggregation of AGD as long as data with ground truth, i.e., precise information on PD and SD, is available. However, synthetic data should be used with caution, as it entails an increased risk of overfitting.

Contents

CHAPTER 1 Introduction

1.1 Background

Train delays are not only inconvenient for passengers but also lead to a growth in costs for operators. They are frustrating for the customers, due to an increase in travel time and missed connections. Operators want to avoid delays since they lead to financial losses through inefficient resource allocation, increased staff hours, and compensation claims. Furthermore, a discontentment of the passengers may result in a decrease in ridership. In the last years, train delays and their impact on passenger satisfaction have been widely discussed in the media. As the German news outlet Tagesschau reports, in 2023, nearly every third passenger arrived with a delay, showing a decline in punctuality compared to previous years when the punctuality rate was around 80% [1]. Additionally, delays in freight services can disrupt supply chains. This might lead to significant financial losses and inefficiencies in logistics. In other words, there are many reasons why train operators are interested in effectively reducing delays.

In order to effectively manage delays, it is crucial to distinguish between Primary Delays (PD) and Secondary Delays (SD) [2]. PD are defined as delays, that are caused by external factors to the train network. They can be triggered by, e.g., weather conditions, signal malfunctions, emergencies with sick passengers, or rails blocked by trees. That means PD would occur even if there were no other trains in the network. Delays that are caused by internal factors are called SD. Sources of SD are events like tracks being blocked by another train or delayed resources (e.g. delayed traction unit). So SD are delays that are caused by other services.

Interestingly, the majority of train operators fail to distinguish between these two categories of delays. They typically collect the sum of PD and SD as *delays*, which will, from now on, be defined as Aggregated Delays (AGD). To reduce the overall amount of delays, it is crucial to distinguish between PD and SD, because SD can be minimized through timetable optimization alone, while PD can primarily be reduced by improvements in infrastructure.

Despite efforts to minimize delays, certain PD, such as a tree on the tracks or an emergency involving a passenger, may be unavoidable. Other PD can be limited at the micro-level such as better infrastructure or improved operating equipment. In contrast, SD can be minimized through more robust planning. This makes it financially more attractive to primarily focus on minimizing SD rather than PD.

The main aim of this work is to develop a model that can disaggregate AGD into PD and SD, enabling more effective timetable planning and delay management strategies. For this study, only unlabeled AGD data of Austrian Federal Railways (OEBB) is available, which means it is not known to what part a delay is PD or SD.

A disaggregation model will be developed by using a previously developed agent-based simulation model [3] of the Austrian railway network. The simulation model takes train schedules and PD as input and computes the resulting SD.

Within this work, a data-driven approach is chosen to address the disaggregation problem at hand. Using the agent-based simulation model, synthetic data will be created for different PD inputs. In the next step, a Graph Neural Network (GNN) will be trained on the generated data with the task of disaggregating the AGD into PD and SD. Finally, the trained model will be evaluated using historical data, which is synonymous with real-world data in this thesis.

1.2 Problem statement

1.2.1 Train Delay Definitions

Primary Delays are delays, that are caused by external influences on the railway system. Examples are disruptions in infrastructure, such as fallen trees on the tracks, severe weather conditions, or technical issues. These primary delays also cover setbacks, such as extended stops due to high passenger numbers in passenger transport or delays in loading processes for freight transport.

Secondary Delays result from causes within the railway network. These occur when a train cannot depart because it lacks a required train driver or when its planned route is blocked by another train. Therefore, SD result from interactions among trains in the network. These delays would not have happened if the affected trains were running independently.

1.2.2 Problem Definition

The objective of this study is to develop a model capable of breaking down the historical AGD of OEBB into PD and SD. This presents a significant challenge, as no ground truth PD and SD values are available. Therefore, it was decided to approach this problem by training a GNN using synthetic data and then later applying it to real-world data. The synthetic delay data will be created using an agent-based model developed by dwh GmbH, which simulates the railway traffic patterns in Austria. The model will be used to generate synthetic SD data for different sampled PD inputs. Subsequently, a GNN will be trained to disaggregate the synthetic data into PD and SD. Finally, this trained GNN will be assessed using the real-world AGD from OEBB by using a simulation-based validation approach.

The problem to be addressed can be defined as follows: $\vec{\mathbf{X}}$ represents the PD, and $\vec{\mathbf{Y}}$ represents the SD in a vectorized form for all trains for a full day. Each row represents the delay for a train at a checkpoint during its journey. The AGD $\vec{\mathbf{Z}}$ can therefore be computed by simply summing up $\vec{\mathbf{X}}$ and $\vec{\mathbf{Y}}$. Using the available agent-based model, $\vec{\mathbf{Y}}$ can be computed as $\mathbf{f}_{\text{agent}}(\vec{\mathbf{X}}) = \vec{\mathbf{Y}}$. Consequently, the overall delay $\vec{\mathbf{Z}}$ can be obtained by:

$$\vec{\mathbf{X}} + \vec{\mathbf{Y}} = \vec{\mathbf{X}} + \mathbf{f}_{\text{agent}}(\vec{\mathbf{X}}) = \vec{\mathbf{Z}} \tag{1.1}$$

Real-world data is only available for $\vec{\mathbf{Z}}$, which is called $\vec{\mathbf{Z}}_{\text{real-world}}$, but the variables of interest are $\vec{\mathbf{X}}$ and $\vec{\mathbf{Y}}$. The goal is to build a model, that can compute $\mathbf{f}_{\text{GNN}}(\vec{\mathbf{Z}}_{\text{real-world}}) = \tilde{\vec{\mathbf{X}}}$. This model allows it to disaggregate the historical delay data into PD and SD. As a result $\tilde{\vec{\mathbf{Z}}}$ can be computed with:

$$\tilde{\vec{\mathbf{X}}} + \mathbf{f}_{\text{agent}}(\tilde{\vec{\mathbf{X}}}) = \tilde{\vec{\mathbf{Z}}} \tag{1.2}$$

The performance of the model can be evaluated by calculating the residual, denoted as $\vec{r}$, which represents the difference between the estimated values and the historical values:

$$\vec{r} = \left| \tilde{\vec{\mathbf{Z}}} - \vec{\mathbf{Z}}_{\text{real-world}} \right| \tag{1.3}$$

For the cases, where synthetic data of PD and SD is available, the model can be evaluated with the following Error $\vec{\epsilon}$:

$$\vec{\epsilon} = \left| \tilde{\vec{\mathbf{X}}} - \vec{\mathbf{X}} \right| \tag{1.4}$$

1.3 Research Questions

This thesis aims to answer the following three research questions:

1. **Parameterization of the Simulation Model:**
   - From what distributions should PD be sampled, and what parameters should be used, to calibrate the agent-based train simulation model to reflect real-world delay distributions?
2. **Performance on Synthetic Data:**

- How effective are GNN in disaggregating train delays into PD and SD components for synthetic train delay data generated with the agent-based train simulation model?

3. **Performance on Real-World Data:**
    - How well does the GNN, trained on the synthetic data, generalize to real-world data for the task of discerning train delays into PD and SD?

1.4 Objectives

This thesis is of an exploratory nature. The goal of this study is to develop a method for disaggregating AGD into PD and SD. This problem will be tackled using a data-driven approach, specifically a GNN. This work has several objectives:

1. **Literature Review:**
    - Within the literature review, existing data disaggregation methods will be investigated.
    - It will be researched how GNN architectures are currently used in the railway domain and what architecture fits the tasks of this study.
    - Furthermore, the stochastic distribution of PD in railway networks will be researched. This is done since PD serves as an input to the available agent-based train network simulation model. In order to create realistic data, it is crucial to sample from representative probability distributions.
2. **Data Creation:**
    - It is aimed to create data, that has a similar structure to the real-world data. To achieve that, the agent-based model will be calibrated.
    - Furthermore, a method that allows the measuring of the similarity between synthetic data and real-world data has to be developed. The created data should capture a wide range of scenarios, to help the model generalize effectively.
3. **Model Development:**
    - The first objective in the model development phase is to transform the railway data into a suitable graph structure for GNN training.
    - Another goal is to implement a GNN architecture, which solves the disaggregation task effectively.
    - Furthermore, a goal is to identify performance metrics that optimally reflect the progress in training the GNN.
4. **Evaluating the Model on Real-world Data:**

- The available real-world data lacks ground truth (only AGD is available without exact SD and PD details). Therefore, an additional objective is to develop a method for effective validation of the GNN on this data. The aim is to use the available agent-based model for the validation of the results without ground truth. The PD predictions of the historical data can be fed into this model. Subsequently, the resulting AGD can be compared with the actual AGD.

1.5 Significance of the Study

As previously mentioned in the background section, the disaggregation of delay data into PD and SD is of great interest to operators of transportation companies. Within this section, several reasons why this study is significant will be listed.

- **Overall importance**: A reliable disaggregation model would allow it to better understand the causes and consequences of train delays. The main advantage would be the possibility to plan timetables that are more robust against SD, resulting in a decrease in overall AGD.

- **Lack of available models**: In the Literature Review (Chapter 2), it is shown that to this date, no such disaggregation model exists, which highlights the importance of working on this problem.

- **Economic impact**: Reducing the SD will lower the costs of OEBB, due to more punctual passenger and freight services. These cost reductions are driven by fewer financial losses from inefficient resource allocation, reduced staff hours, and a decrease in compensation claims.

- **Passenger experience**: More reliable train schedules will also improve the passenger experience and therefore increase customer satisfaction. This might lead to more people using railway services.

- **Applicability to other rail networks**: The research outcomes are not only valid for the Austrian train system but can also be applied to other railway networks if the model is fine-tuned to the respective data.

- **Interdisciplinary importance**: Overall, the concept of data disaggregation is important in multiple domains. Hence, the insights from this research can add to the existing scientific knowledge regarding disaggregation tasks. Moreover, this work exemplifies the effective integration of simulation techniques with Artificial Neural Network (ANN), a methodological approach that, despite its growing relevance, is seldom employed in current research.

1.6 Overview of the Thesis Structure

The following, a short overview of the thesis structure will be given:

- **Chapter 1: Introduction** - The introduction provides the essential background information needed to understand the research problem. Furthermore, the research questions and objectives are outlined. It further explains the significance of the study for railway operators and the scientific community.
- **Chapter 2: Literature Review** - In the literature review, the available research regarding the distribution of PD is explored. It also elaborates on what data disaggregation models are used in the railway domain and other areas. Apart from that, the use of GNN in train networks is examined.
- **Chapter 3: Graph Neural Networks** - In this chapter, the theory of GNN is described, focusing on Gated Graph Convolutional Networks (GatedGCN), as this type of ANN is used as a key component in this study.
- **Chapter 4: Synthetic Data Generation Using the Agent-Based Model** - Within this chapter, the historical data is analyzed thoroughly, and the synthetic data generation process is described.
- **Chapter 5: Data Preprocessing and Model Implementation** - In this chapter, the preprocessing of the data, the transformation to a graph structure, and the model setup are described.
- **Chapter 6: Results** - This chapter presents results of the developed GNN model. The GNN is evaluated on the synthetic data and the real-world data.
- **Chapter 7: Discussion** - In the discussion, the results obtained by the experiments are reviewed and interpreted. In addition, the contributions to the railway domain are discussed, and the limitations are elaborated.
- **Chapter 8: Conclusion** - In the last chapter, the key findings are discussed, and a final conclusion is drawn.
- **Appendix A: Supplementary Materials** - Supplementary information for the other chapters is included in the appendix.

CHAPTER 2

Literature Review

2.1 Literature Research Methodology

The literature research was conducted in a methodological manner. Several popular literature portals were examined for relevant literature. These were Google Scholar, IEEE Xplore, and ACM Digital Library. A wide variety of key terms were used, which were also varied using different synonyms and Boolean combination operators. Among them were: “disaggregation of train delays,” “propagation of train delays,” “data disaggregation using artificial neural networks,” “primary and secondary train delays,” “distribution of primary delays in railway networks,” “statistical analysis of primary delays,” “graph neural networks and railway networks,” “data disaggregation using graph neural networks,” and “simulation-based disaggregation.”

For SD, the synonyms “knock-on,” “consecutive,” “reactionary,” “flow-on,” and “propagated delay” were also used during the literature search, as they were found in the available literature. For PD, the synonyms “original delays,” “entrance delays,” “source delays,” and “initial delay” were used. The term “disaggregation” was also substituted with “decomposition.”

The main focus of the literature research was on the period from 2014 to 2024.

2.2 Overview

This literature review will give an overview of what was considered the most relevant publications with regard to the previously formulated research questions. First, studies that have had very similar target questions, such as finding delay causes in the railway domain, will be introduced. Next, different studies about the distribution of the duration of PD will be investigated. Then it will be elaborated on how GNN has been utilized so far in train networks and how well it has performed in those studies. In the later sections, the challenges of GNN for the given research questions are discussed, and data disaggregation methods in other domains are briefly introduced. The goal of this section is to give an overview of the state-of-the-art and provide rationales for the method that has been chosen to tackle the research questions.

2.3 Disaggregating Train Delays into Primary and Secondary Delays

To this point, there is very little research regarding the disaggregation of AGD into PD and SD. In [2], the delay discerning task is indirectly tackled. The idea was to build a machine-learning model that predicts the additional delay for each train at each stop. The input features are divided into two categories: features that are causes of PD (weather, day of the week, season, vehicle type) and features that can be viewed as causes of SD (delays of other trains). As the machine learning model predicts the delay at a given station, the extent to which each feature contributed to the prediction can be analyzed. This is done using Explainable Artificial Intelligence (XAI), specifically using a method called SHAP (SHapley Additive exPlanations) values. This approach allows an effective decomposition of the prediction. The study can be considered the publication that tackles questions most similar to the research questions in this work. Nevertheless, it does not propose a final model where the inputs are the actual delays and the outputs are the disaggregation. It must be considered that there are also other more complex SD sources in railway systems, which are not considered in the study, such as delayed personnel or delayed traction units.

In [4], the goal was not directly to discern AGD, but rather to find single sources of delays via a backtracking algorithm called *Critical Path Search*. In other words, it can output the PD for a given SD. However, its limitation is that it classifies delays as either SD or PD in a binary manner. Also, the focus of this publication is on single sources of PD. Therefore, in most scenarios, the algorithm will categorize delays as SD, with only a few identified as PD causes. This method is also reviewed in [5], where it is compared to another method of source estimation for propagation processes in complex networks. Similarly, [6] introduces a visualization-based technique to identify delay propagation routes.

In [7], the proportions of PD and SD for a part of the Swedish train network were computed. These proportions were only estimated for the whole train network but not for delays occurring for single trains at different stations.

Based on the listed literature, it can be concluded that the available literature for the disaggregation of AGD is very limited, and there is a need for further research. The study closest to the research questions is [2]. While this study provides good ideas and a valuable approach, there are two aspects in which it differs from the research questions tackled in this work. First, in the study, the researchers work on one dataset, in contrast to training the model on synthetic data and then applying it to real-world data. Second, no direct disaggregation model was proposed. Instead, a delay prediction model was developed, which shows to what extent each delay is caused by PD and SD sources. On the other hand, [4] suggests a backtracking algorithm to find single sources of PD. While this is a valuable approach, the goal of this work is to follow a data-driven approach. To this point, no data-driven disaggregation model for AGD has been published. Therefore, it can be concluded that there is a research gap worth investigating.

2.4 Stochastic Distribution of Primary Delays in Railway Networks

Selecting the appropriate distribution and parameterization of PD is essential for accurate modeling. The distribution of delay durations associated with PD is particularly interesting. To gain insights into this aspect, the literature review focuses on studies investigating the statistical properties and distributions of PD in railway networks. Unfortunately, very few publications on the distributions of PD are available.

[8] provides a detailed research paper on various distribution statistics of PD on a high-speed railway train track between Wuhan and Guangzhou in China. The study investigates temporal, spatial, and durational distributions of PD. A goodness-of-fit test was carried out for several distributions to determine which distribution most accurately approximates the time span of PD. It was concluded that the duration of PD follows a log-normal distribution and also published the estimated parameters.

In [9], the different sources that cause PD were investigated. Overall, seven different causes, e.g., weather or failure of tracks, and distributions that approximate the duration distributions best were listed. The study concluded that log-normal distributions are most suitable to model the duration of all seven delay sources.

In [10], it was found that the distribution of PD can best be described by a negative exponential distribution. It has to be noted that first, the data in this study is very limited, and second, it is from 1974. Still, it is mentioned due to limited research on this specific field.

2.5 Graph Neural Networks

2.5.1 Graph Neural Networks in Railway Networks

This section provides an overview of how GNN have been applied to railway networks in past literature.

In [11], a GNN is used to predict train delays in railway networks. A so-called heterogeneous GNN was used, which allows nodes with different attributes as input. This type of GNN made it possible to design a graph with nodes such as trains and stations, allowing complex graph structures. Among other things, the study shows that GNN are capable of capturing the propagation mechanisms in train networks. This is highly relevant for this thesis since it is one of only a few papers so far where GNN has been applied within the railway domain. Furthermore, the proposed model outperforms existing delay prediction methods.

In [12], a GNN, more specifically, a Spatial-Temporal Graph Convolutional Network, was used to predict future delays of trains for a subset of the British rail network. The researchers show that this model is superior to other models, namely Multilayer Perceptron and linear regression. The proposed model effectively predicts the delays by capturing the non-linear dependencies within the network. Here again, it is shown that GNN is a suitable method to better understand and model the delay propagation of trains.

To support dispatchers in their decision-making, [13] outlines a GNN that was developed, aiming to give an overview of the network congestion and delay patterns. In the paper, the researchers introduce a deep learning framework called Train Spatio-Temporal Graph Convolutional Network. The model forecasts the total number of delayed trains over a period at different stations. It was trained on data from the China Railway Passenger Ticket System, specifically on a subset of railway data around the city of Zhengzhou. In the study, the model outperforms all comparison models.

A novel Graph Attention Network model to predict and explain train delay propagation in railway networks is proposed in [14]. The study strongly focuses on the interpretability aspect of delay propagation within train networks. The model allows different weights to be assigned to the edges, making the influences of different factors like headway times or arrival delays visible. This provides a deep insight into the mechanisms of delay propagation. The study used a subset of the Dutch railway network, specifically the mainline from Amsterdam to Utrecht. The novel architecture outperformed existing models such as Bayesian networks and Markov models. The study underscores the effectiveness of GNN to improve the prediction of train delays and their interpretability.

A study [15] investigates the use of GNN for predicting delays within the Chinese high-speed train network. The proposed model uses graph community neural networks and time-series fuzzy decision trees. It not only led to high accuracy but also delivered interpretable results. The model outperformed seven state-of-the-art models.

To get a general overview of how GNN can be used for traffic forecasting problems, [16] is a useful reference. In this review, studies from all kinds of transportation domains, e.g., subway systems or taxis, are compared and discussed. The paper can be seen as a valuable starting point for researchers who want to apply GNN to tasks related to transportation systems.

Overall, it can be seen that applying GNN in the railway domain is a rather new development, with the first publications arising around 2020. Nevertheless, it seems like a highly promising approach, as the mentioned studies report high accuracies and often interpretable results. It can be concluded that GNN are able to capture the non-linear delay propagation mechanisms within railway networks, especially because graphs are a highly suitable data structure for train data. From this finding, the overall idea to use GNN for tackling the research problem of disaggregating AGD into PD and SD was developed.

2.5.2 Data Disaggregation in Nonintrusive Load Monitoring

Data disaggregation is relatively well-researched in the electricity domain, more specifically in Nonintrusive Load Monitoring (NILM). This technique discerns households' aggregated electricity data into the consumption patterns of individual devices. It provides a cost-effective and convenient alternative to the traditional method of gaining energy consumption data from single devices, which involves equipping them with electricity monitors to record the relevant data.

NILM is a well-established research field resulting in numerous studies that suggest and compare different disaggregation methods. Although railway network data generally has a pretty different structure, it still seems reasonable to briefly discuss data disaggregation in the electricity domain.

The review [17] gives a broad overview of the advances in NILM. Relevant for the present research is the fact that it shows that ANN performs very well in the area of data disaggregation, with the downside of high computational cost during training and the need for large datasets.

Interestingly, there was one study [18] published in 2024, which leverages GNN to disaggregate electricity data. The paper suggests that the architecture is highly efficient at its task, outperforming other models. Nevertheless, this gain cannot be directly transferred to the elaborated research questions, as first, the relevant patterns in the electricity data are most probably very different from delay propagation patterns in railway data, and second, the network is trained and tested on one dataset, meaning it is not trained on synthetic data and then applied to another dataset.

In short, studies from NILM show that ANN can be effectively applied for data disaggregation tasks.

2.5.3 Graph Neural Networks for Long-Range Tasks

The graphs that will be the input to the GNN can be considered very large since they represent the whole railway traffic of a day within Austria, which has around more than 9700 km of railway tracks [19]. Based on the fact that delays are often propagated through several trains, it can be assumed that to effectively model train delay transmission, information needs to pass through multiple nodes, often spanning distances greater than two nodes. For most GNN architectures, long-range message-passing requires many layers, as each layer only leads to information propagation to a node's neighbors (Chapter 3). Apart from very high computational demands, a high number of layers also leads to over-squashing. This phenomenon is thoroughly explained in [20]. Nevertheless, due to the graph structure and graph size, long-range message-passing will be required for the research problem. Therefore, GNN architectures that perform effectively on long-range tasks are discussed.

To test the ability of GNN architectures to model long-range dependencies, the *Long-Range Graph Benchmark* [21] was established. It can be seen that GatedGCN perform universally well on a wide range of tasks, despite being one of the baseline models. The proposed transformer-based approaches work on fully connected graphs, therefore having a complexity of $O(N^2)$. For that reason, the computational costs would be extremely high for graphs of the size faced in this research. GatedGCN are comparatively cheap to train, while still providing high accuracy rates, in some cases even exceeding transformer-based approaches. In [22], where the long-range benchmark was reassessed through more thorough fine-tuning, GatedGCN performed even better. They also offer the benefit of edge weighting compared to usual Graph Convolutional Networks (GCN), making them more interpretable. This study will use GatedGCN as layers in the GNN due to their interpretability and relatively low computational costs (compared to transformer-based methods), while still proving to be effective at solving long-range tasks.

One approach to improve long-range message-passing without adding a high number of layers is hierarchical message-passing. This approach, as described in [23], suggests adding supergraphs at different hierarchy levels. A supergraph is a simplified representation of the original graph, where similar nodes are grouped into clusters and each cluster is represented by a single supernode. This supernode encapsulates the aggregated information of all the nodes in its cluster. Several hierarchy levels can be added, depending on the requirements. Message-passing happens within the original graph and the supergraphs, but also top-down and bottom-up. A hierarchical message-passing approach was also taken in [24], where a supergraph was created using a specific pooling algorithm. The idea of using hierarchical structures influenced the graph design in this work. To enable long-range message passing, it was decided to add a modified type of supernodes for single trains, as described in Section 5.4.

In conclusion, there is plenty of research on how long-range message-passing can be accomplished in GNN. To get an overview of current state-of-the-art architectures, it is recommended to view updated results on the *Long-Range Graph Benchmark*.

2.6 Overcoming Challenges in Out-of-Distribution Learning

ANN can be powerful in many areas. However, in most cases, large amounts of high-quality labeled data are necessary to train high-performing networks, which is often a bottleneck. Therefore, using some form of artificial data in deep learning is common practice, as described in [25]. Simulations are one of many methods of creating synthetic data. Here, [26] is an example, providing a guideline on different strategies to use synthetic simulated data in the construction domain. [27] provides a case where data generated by an agent-based model was successfully used for deep learning training.

In [28], it is elaborated on how simulators might be a crucial step in the advancements of deep learning, as large quantities of labeled data can be generated with simulations. This data can then be used, for example, for pretraining neural networks. The study discusses that ANN often have a low performance if they are trained on artificial data and then tested on real-world data. This effect usually happens due to dissimilar feature distributions within the different datasets. Therefore, developing methods to close the gap between the varying feature representations is crucial.

In this study, even in cases of rigorous calibrations of the available agent-based model, it can be expected that the output will still differ significantly from real-world data due to simplifications based on different model assumptions, missing data points, suboptimal parameter settings, etc. This potential difference between synthetic and real-world data must be considered since it might pose the risk that the model overfits the synthetic data and consequently predicts poorly on the real-world data. The methods to improve Out-of-Distribution (OOD) generalization can be separated into three categories as done in [29]:

- **Data** - varying and modifying the input graphs
- **Learning strategy** - implementing optimization objectives and constraints
- **Model** - adapting the GNN architecture for better OOD performance

A comprehensive overview of these methods can be found in [29].

In conclusion, the literature review showed that there are several critical gaps in the realm of researching railway systems that are worth investigating. The aim of this thesis is to fill these gaps by using a GNN to discern AGD into PD and SD. This will be approached by training the model on synthetic data generated by an agent-based model and subsequently validating it on real-world data.

CHAPTER 3

Graph Neural Networks

As GNN are a central part of the methodology of this study, the theory behind them will be elaborated on in this chapter. First, the principles of GNN, like message-passing, are described. Next, specific variants as GatedGCN will be explained. The chapter is primarily based on the first publication of GNN [30] and an introductory work [31]. Furthermore, [32] was used as a source for GatedGCN. As this study focuses on the application of GNN, this chapter is intended only to provide an overview of the mathematical principles. For more detailed information, consulting the primary sources is recommended.

3.1 Introduction to Graph Neural Networks

Graphs are widely used data structures to represent complex relationships and interactions in a variety of scenarios. They can be represented as $G = (V, E)$, where V denotes the set of nodes and E represents the set of edges. Furthermore, $|V| = N$ depicts the number of nodes, and $A \in \mathbb{R}^{N \times N}$ the adjacency matrix that encodes the connections between the nodes.

Article [33] outlines various domains where GNN are effectively applied to solve diverse problems, such as in chemistry, social networks, web page ranking, and physical networks. Additionally, it elaborates on various prediction tasks, including node and graph classification, as well as link prediction, where the goal is to predict connections between two nodes.

These networks aim to learn efficient node representations that encode both the features of individual nodes and their relationships with their neighbors. Through an iterative learning process about the node, its features, and how it belongs in the whole graph, the GNN learns complex relational information within the graph.

High-Level Structure of Graph Neural Networks

The key element of a GNN is the message-passing layer. This part of the network enables information from one node to be passed to its neighbors. A GNN usually consists of several such layers since the number of layers controls how far information can be passed within the network. The features of a node are stored in a vectorized form, which is called node representation. These feature embeddings change with each message-passing layer.

The overall structure of a GNN can be abstracted as:

1. **Input Layer**: Each node i of the graph has an initial feature vector $\mathbf{h}_i^{(0)}$.
2. **Hidden Layers**: Each hidden layer k consists of:
   - **Message-Passing**: Each node computes a message that will be sent to its neighbors.
   - **Aggregation**: Each node aggregates the messages received.
   - **Node Update**: Each node updates its representation based on its previous embedding and the aggregated messages.
3. **Output Layer**: The last layer simply outputs the representation of each node. It can be used for a variety of tasks like node classification or link prediction.

How far a message can be passed within a GNN depends on the number of layers, also referred to as iterations k. Each layer in the model leads to an additional message-passing step. If a message is passed from one node to its neighbors, it is called a hop. In cases where it is necessary to, e.g., pass messages to nodes with a path distance of four, it would be necessary to add at least four layers to the GNN to create a 4-hop network.

3.1.1 Message-Passing

As mentioned, the message-passing mechanism is a key element of GNN. Within this section, the three main steps of message-passing,

1. Message computation,
2. Message aggregation, and
3. Node update

will be explained in more detail.

Message Computation

In the first step, the message computation, every node i computes messages which are then sent to the neighbor nodes j. A message $\mathbf{m}_{ij}^{(k+1)}$ is usually computed based on both nodes' current representations and the features of the outgoing edges. $\mathbf{h}_i^{(k)}$ and $\mathbf{h}_j^{(k)}$ represent the features of nodes i and j for iteration k. The features of the edges that connect the nodes are described as $\mathbf{e}_{ij}$. Based on this, the function that computes the messages, MESSAGE, is defined as

$$\mathbf{m}_{ij}^{(k+1)} = \text{MESSAGE}(\mathbf{h}_i^{(k)}, \mathbf{h}_j^{(k)}, \mathbf{e}_{ij}), \tag{3.1}$$

where $\mathbf{m}_{ij}^{(k+1)}$ represents the message passed from node i to node j at iteration $k+1$.

Message Aggregation

After the different messages are computed, the message aggregation step follows. During this step, each node i aggregates the incoming messages from its neighbors $\mathcal{N}(i)$. To compute the aggregated message $\mathbf{m}_i^{(k)}$ for node i at iteration k, an aggregation function AGGREGATE is defined. This function combines the messages into a single representation that captures the information of the nodes in the neighborhood. Mathematically, the message aggregation can be described as

$$\mathbf{m}_i^{(k)} = \text{AGGREGATE}\left(\mathbf{m}_{ji}^{(k)} \mid j \in \mathcal{N}(i)\}\right). \tag{3.2}$$

There are different design choices for the message aggregation. A simple yet often-used function is the sum aggregation, which can be denoted as

$$\mathbf{m}_i^{(k)} = \sum_{j \in \mathcal{N}(i)} \mathbf{m}_{ji}^{(k)}. \tag{3.3}$$

Node Update

After the messages are aggregated, the node update step follows, which can be described as

$$\mathbf{h}_i^{(k+1)} = \text{UPDATE}(\mathbf{h}_i^{(k)}, \mathbf{m}_i^{(k)}). \tag{3.4}$$

During this step, each node i updates its representation based on the aggregated message and its embedding before that timestep. Usually, a neural network layer followed by a non-linear activation function is used to perform this step.

The mechanisms of message-passing build the foundation of GNN. To improve performance and functionality and offer solutions to different problems, a large number of variants and extensions of the basic framework have been developed. One notable and widely used extension is GCN.

Graph Convolutional Networks

GCN apply the concept of Convolutional Neural Networks (CNN), as introduced in [34], to the area of GNN. In other words, GCN utilize convolutional operations on graphs. The convolutions and, consequently, the node update can be efficiently performed using the normalized adjacency matrix. In contrast to GNN, GCN reduces the feature vector size with each iteration.

The update function of the GCN can be expressed as

$$\mathbf{h}_i^{(k+1)} = \sigma\left(\mathbf{W}\mathbf{h}_i^{(k)} + \mathbf{m}_i^{(k)}\right), \tag{3.5}$$

where $\mathbf{W}$ represents a learnable weight matrix and σ a non-linear activation function.

3.2 Variants and Extensions of Graph Neural Networks

While GCN provide a robust extension of traditional GNN, they come with several notable limitations. One major limitation is the equal weighting of all neighboring nodes, which might prevent the network from learning certain important relationships. Additionally, GCN can suffer from over-smoothing. This describes the effect where node representations become indistinguishable as the number of layers increases. GatedGCN have been proposed to address these issues, as described in [32]. Through gating mechanisms, which control the flow of information between nodes, this architecture enhances the capabilities of GCN. These gating mechanisms allow information to be selectively passed through the network based on the relevance of the connections. This feature allows the capture of more complex relationships within graphs.

The influence of the neighbors of a node on its new representation is controlled by the gating mechanism. This mechanism leverages the edge gates η_{ij} to filter only the important information of each connection. The edge gates can be computed with

$$\eta_{ij} = \sigma\left(\mathbf{A}\mathbf{h}_i + \mathbf{B}\mathbf{h}_j\right), \tag{3.6}$$

where $\mathbf{A}$ and $\mathbf{B}$ are learnable weight matrices.

The GatedGCN update rule can be expressed as

$$\mathbf{h}_i^{(k+1)} = \sigma\left(\mathbf{U}^{(k)}\mathbf{h}_i^{(k)} + \sum_{j\in\mathcal{N}(i)} \eta_{ij} \odot \mathbf{V}^{(k)}\mathbf{h}_j^{(k)}\right), \tag{3.7}$$

where $\mathbf{U}^{(k)}$ and $\mathbf{V}^{(k)}$ represent learnable weight matrices. The symbol $\odot$ represents the Hadamard product.

CHAPTER 4

Synthetic Data Generation Using the Agent-Based Model

This chapter describes how the agent-based model was calibrated to produce synthetic railway delay data. By leveraging historical data and domain knowledge, the aim was to identify optimal parameters that enable the available agent-based model to generate a realistic dataset capturing the complex dynamics of railway operations.

The main objectives of this chapter are:

- Explore and analyze historical delay data to gain insights into delay distributions in the Austrian rail network.
- Review experiments conducted to determine optimal parameters for sampling delays, ensuring a close match with real-world patterns.
- Evaluate the generated synthetic data and assess its plausibility compared to real-world data.
- Discuss the limitations of the current approach and identify areas for future improvement.

4.1 Data exploration of Historical Delay Data

4.1.1 Dataset Description

First, the data provided by OEBB was analyzed. The dataset encompassed a single day's train traffic records, including both passenger and freight trains within the Austrian rail network. The primary goal of this section is to deliver an understanding of how the different delays are distributed. Hence, the analysis concentrates on features related to delays. All other features are described in Section 5.2.

To ensure data confidentiality, several data scales were normalized. This preprocessing allows pattern analysis while anonymizing the data.

4.1.2 Analysis of Train Arrival Patterns

Figure 4.1 shows the train arrivals at stations for passenger trains during one day. This histogram displays the normalized number of trains present in the railway network at various times, revealing two peaks throughout the day, indicating the rush hours.

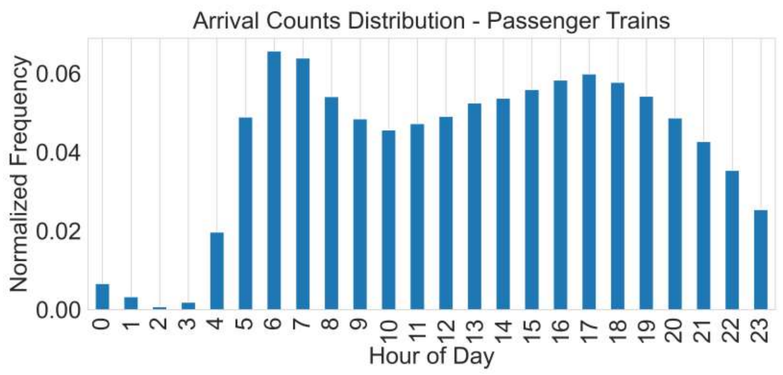


Figure 4.1: Distribution of Arrival Records for Passenger Trains

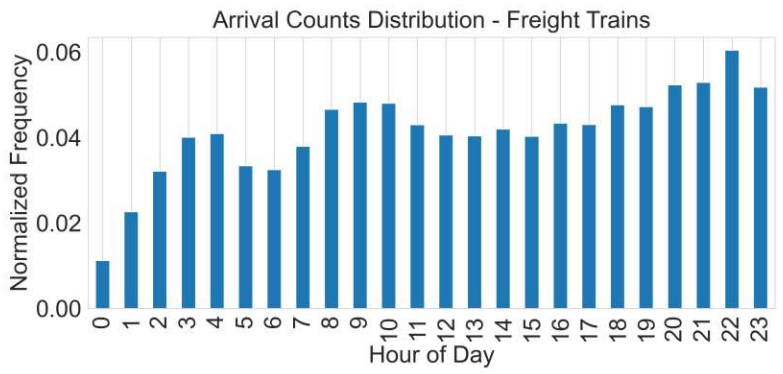


Figure 4.2: Distribution of Arrival Records for Freight Trains

Figure 4.2 shows the same type of plot for freight trains. In this case, the train distribution shows a distinct pattern compared to that of the passenger trains.

4.1.3 Analysis of Arrival Delay Durations

This section analyzes the distribution of delay durations. For simplicity, the analysis focuses solely on arrival delays. Figure 4.3 and Figure 4.4 show the arrival delay duration distributions of passenger and freight trains. The plots aim to give an understanding of how different delay duration distributions are depending on the type of train.

To anonymize the data, min-max normalization 4.1.1 was used for a predefined subset of the data. The same minimum and maximum values were used for both the passenger and freight train plots to make the plots comparable.

Definition 4.1.1 (Min-Max Normalization)**.** Min-max normalization is a rescaling technique applied to features to transform their values into a specific range, typically $[0, 1]$. For any given feature x with observed minimum and maximum values denoted as x_{min} and x_{max}, respectively, the normalized value x' is calculated using the following formula:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{4.1}$$

Here, x represents the original value of the feature, x_{min} is the minimum observed value, and x_{max} is the maximum observed value. This normalization technique ensures that the transformed feature x' falls within the range $[0, 1]$, thereby facilitating comparison and analysis while preserving the relative relationships between the original data values.

Figure 4.3 indicates that the majority of passenger trains are punctual or have minimal delays. The distribution of delay length for freight trains significantly differs, as seen in Figure 4.4. A greater proportion of freight trains experience delays compared to passenger trains. Also, the delays are significantly longer than the delays of passenger trains. This is understandable, given that punctuality holds less importance for freight trains than for passenger trains. Freight trains have different operational priorities and logistical complexities associated with cargo transportation.

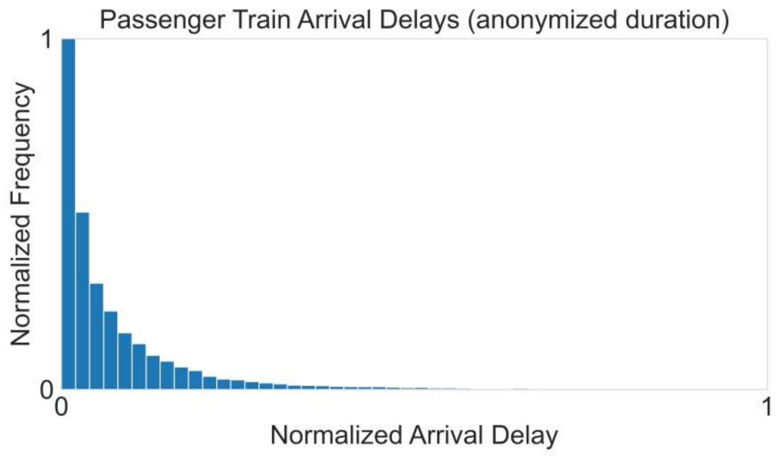


Figure 4.3: Normalized Passenger Train Arrival Delay Durations

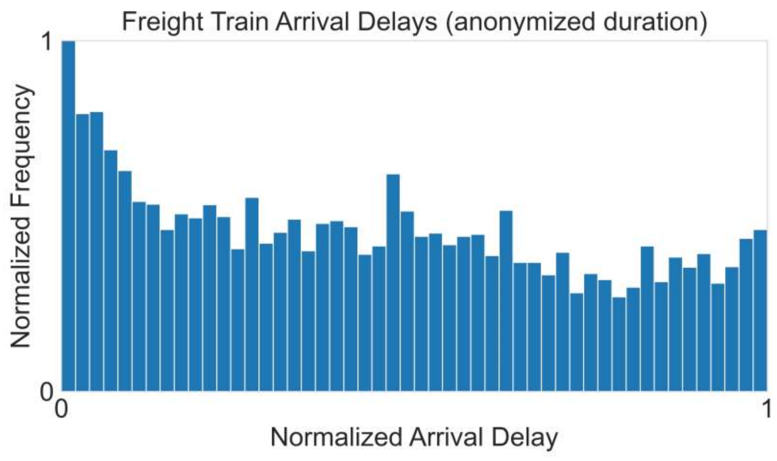


Figure 4.4: Normalized Freight Train Arrival Delay Durations

4.1.4 Analysis of Delay Change

This section elaborates on when delays occur and their magnitudes. To achieve this, the concept of *delay change* and *sequence numbers* are introduced. A *sequence number* is defined as a marker that represents the order of checkpoints a train passes through during its trip, where sequence $= 1$ indicates the start of the trip. Delay change is measured by the variation in delay from one checkpoint to the next, starting with an initial zero delay. If a train starts its journey with a 15-minute delay, the delay change for sequence $= 1$ is noted as 15 minutes. This metric gives a better understanding of when delays happen for single trains and their magnitudes.

Delay changes can have positive or negative values. A positive delay change indicates that a train has an increase in delays, while a negative value shows that a train makes up time.

Figures 4.5 and 4.6 show the delay changes for the same positive predefined range of passenger and freight trains. The values have been normalized to anonymize the data. Both figures employ a logarithmic scale on the y-axis to cover a broad range of frequencies. The plots were included to show the different distributions of delay changes between freight and passenger trains. A clear decrease in frequency as the delay change gets larger can be seen in both plots. It can also be seen that the effect is stronger for passenger trains. Passenger trains focus on staying on time resulting in an effort to adhere to schedules.

4.1.5 Initial Delay Change

As mentioned, it is also important to know when delay changes happen for individual trains. Therefore, the average delay changes across all trains at different sequence numbers were analyzed to get further insights into patterns. It was observed that delays generally offset each other, yet at the initial stop for freight and passenger trains, positive average

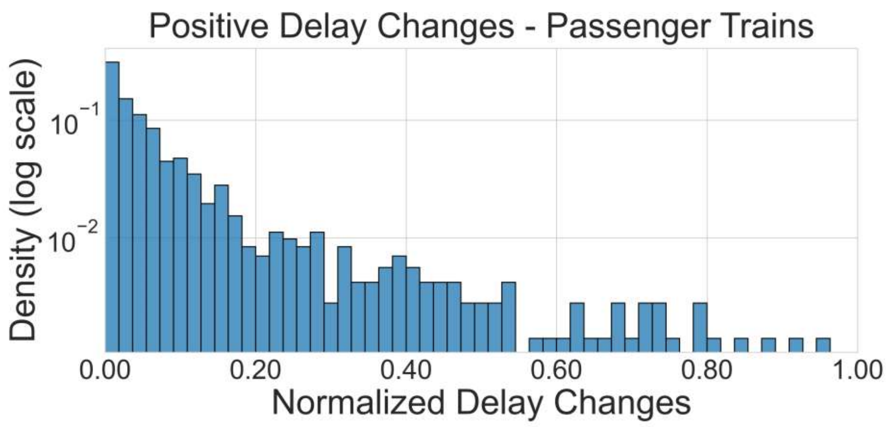


Figure 4.5: Distribution of Delay Changes for Passenger Trains

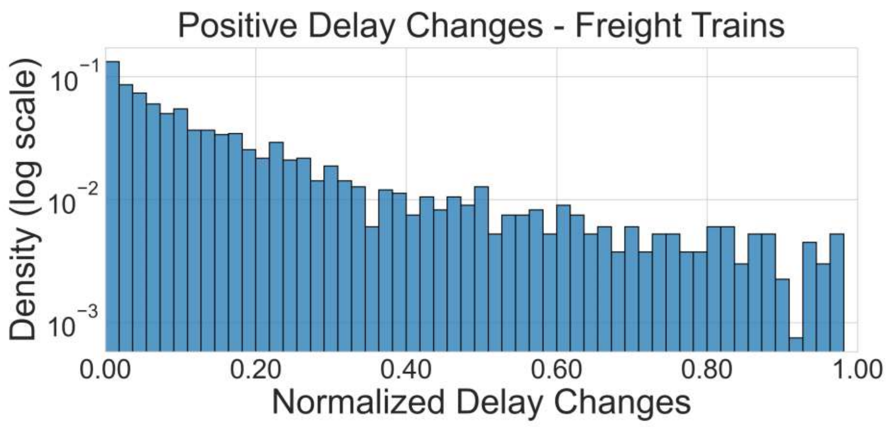


Figure 4.6: Distribution of Delay Changes for Freight Trains

delays were observed. This makes sense since passenger trains should not depart early, so they will either depart punctually or late at the first stop, meaning that there should be no negative delay changes for passenger trains at the start of a trip. In practice, freight trains sometimes start before schedule. That's why the initial delay change of freight trains can also be negative. Nevertheless, the average initial delay change was positive for the analyzed data. This indicates that many freight train delays happen in the beginning.

The initial delays of freight trains were further analyzed since they were a manifold of the initial delays of passenger trains. It was observed that a small fraction of freight trains start before schedule. A majority of the freight trains start later than scheduled. Based on expert knowledge, most of those delays are caused by:

- The lower prioritization compared to passenger traffic
- The delay of resources necessary for operating the train

Thus, it is assumed that a majority of these delays are SD. A sensitivity analysis A.1 was conducted to test whether the agent-based model accurately reproduces these expected SD. The analyses indicate that these SD are not adequately represented in the simulation to the extent anticipated by experts, highlighting a limitation in the model.

4.2 Methodology of Data Generation

In this section, the methodology that was used for the creation of data is outlined.

4.2.1 Approaches for the Primary Delay Sampling

Two methods were tested for generating synthetic data. Only the second will be elaborated in detail in this thesis, as it proved to be superior. The first approach was based on the findings from the literature research in Section 2.4. It was assumed that the PD duration distributions within the Austrian railway network are similar to the distributions suggested in [8]. Therefore, the log-normal distribution parameters suggested in [8] were used for sampling PD durations. A comparison metric to compare the AGD of synthetic data to the real-world data was established. This finally led to the estimation of delay occurrence frequencies and synthetic data. Due to a poor fit of the generated data to the real-world data, this approach was discarded.

The approach that was used for the final data creation is based on the assumption that the distribution of PD duration follows the same distribution as the delay changes in the historical data. This method facilitates not only the easy determination of distribution parameters but also provides starting points for the occurrence frequencies of delays. The exact parameter estimation process is described in this chapter.

In the following section, the assumptions made are first elaborated on. It is then explained from which distributions PD durations are sampled. Next, upper boundaries for PD occurrence frequencies are derived from historical data. After a parameter-finding process through grid search, a plausibility analysis is conducted. The result is a set of feasible distributions, distribution parameters, and delay occurrence frequencies from which PD durations are sampled to create a final dataset.

4.2.2 Assumptions for Primary Delay Distributions

The goal of this sampling strategy was to include the most important factors while keeping the complexity as low as possible. To achieve this, several assumptions about the occurrence of PD and the distribution of PD durations were made.

Assumption 1: The distribution of the PD follows the distribution of the delay changes in the historical data.

Assumption 2: The frequencies of PD occurrences and the distributions of PD durations are assumed to be independent of both location and time.

Assumption 3: Frequencies of PD occurrences and the distributions of PD durations are considered independent of Operational Control Point (OCP) type.

These assumptions allow for the estimation of the necessary parameters to sample PD using the available data. Based on the first assumption, PD duration distributions can be estimated by fitting distributions to the delay change from the historical data. The idea is to fit several distributions and afterward determine the probability $P(d > 0)$ or p_0 of delay occurrence. In mathematical terms, this can be described as estimating the probability $P(d = x)$, where $d = x$ represents the occurrence of a delay change of exactly x minutes:

$$P(d = x) = P(d > 0) \times P(d = x \mid d > 0) = p_0 \times P(X = x) \tag{4.2}$$

Here, $P(d = x \mid d > 0)$ denotes the probability of a delay change being x minutes, given that a delay has occurred. The probability distribution of the delay change durations X is represented as $P(X = x)$. It gives the likelihood that X has the value x if a delay takes place.

The second assumption can be considered to be very restrictive. Nevertheless, it is reasonable since no significant statements about the specific distributions can be made at different times or locations due to the availability of only a single day of data. Here, future improvements are possible if a larger dataset is available, as discussed in the Limitations 4.5.

4.2.3 Distributions for Sampling Primary Delays

Building on the assumptions, the next step was to determine the appropriate distributions for sampling PD delay durations. Overall PD were sampled from four different distributions, which represent various delay scenarios:

- PD for passenger trains at the start of the trip (sequence = 1)
- PD for passenger trains during the trip (sequence > 1)
- PD for freight trains at the start of the trip (sequence = 1)
- PD for freight trains during the trip (sequence > 1)

The decision to model the four scenarios was based on the data exploration (Section 4.1) and the stated assumptions. First, the analysis showed that freight and passenger trains have very different delay change duration distributions. Second, it revealed that the delay change duration distributions at the departure of a train journey are markedly different from those during the actual journey.

The delay change duration distributions are highly skewed. There is a high frequency of occurrences in the low second range, tapering off to a few occurrences in the several-hour range. This skewness made it particularly challenging to fit distributions that accurately capture the entire range of data. To fit distributions, the Kolmogorov-Smirnov Test (K-S Test) 4.2.1 was used.

Definition 4.2.1 (Kolmogorov-Smirnov Statistic (KS))**.** The K-S Test statistic quantifies the maximum difference between the empirical cumulative distribution functions of two samples:

$$D = \sup_{x} |F_n(x) - F_m(x)| \tag{4.3}$$

where F_n and F_m are the empirical cumulative distribution functions of the two samples [35, 36].

A lower K-S Test statistic indicates a closer match between the synthetic and observed delay distributions, implying a more accurate representation of real-world delays.

Distributions were fit to the range of delay change durations of one second to an upper boundary of three hours, using a single distribution. Given the high frequency of delays in the low-second range and the long tail of the data, the K-S Test demonstrated that the Pareto distribution was superior to other distributions (lognormal, Weibull, exponential, gamma) for a majority of cases.

To illustrate this, the following section presents the exact distribution parameters estimated for the four different delay scenarios: passenger trains at the start of the trip, passenger trains during the trip, freight trains at the start of the trip, and freight trains during the trip.

Passenger Train Start

The first scenario focuses on the delays at the start of a passenger train's journey. The following table and plot show the Pareto distribution parameters and the goodness-of-fit statistics for this scenario.

Distribution	Pareto
Parameters	Shape: 1.3908, Location: -0.9476, Scale: 0.9642
D-statistic	0.035
n	3472

Table 4.1: Pareto Distribution Fit for Passenger Train Start Delay Changes

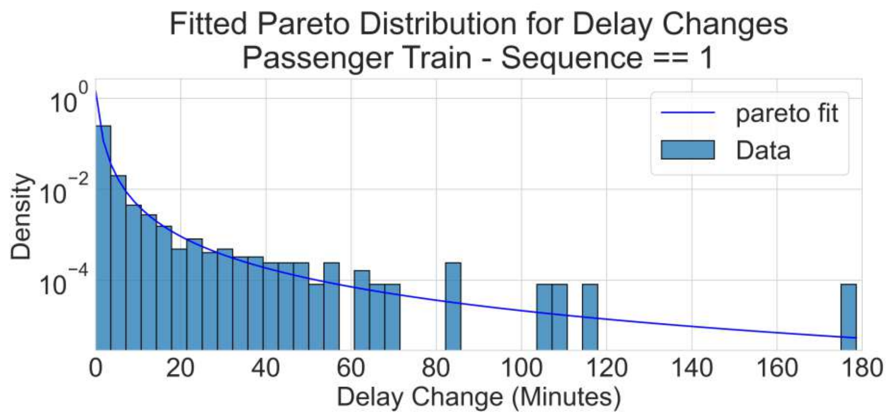


Figure 4.7: Pareto Fit for Passenger Train Start

Passenger Train Journey

The second scenario examines the delays occurring during the journey of passenger trains. The table and plot below present the parameters of the Pareto distribution and the associated goodness-of-fit statistics.

Distribution	Pareto
Parameters	Shape: 2.6440, Location: -0.6330, Scale: 0.6497
D-statistic	0.053
n	81773

Table 4.2: Pareto Distribution Fit for Passenger Train Journey Delay Changes

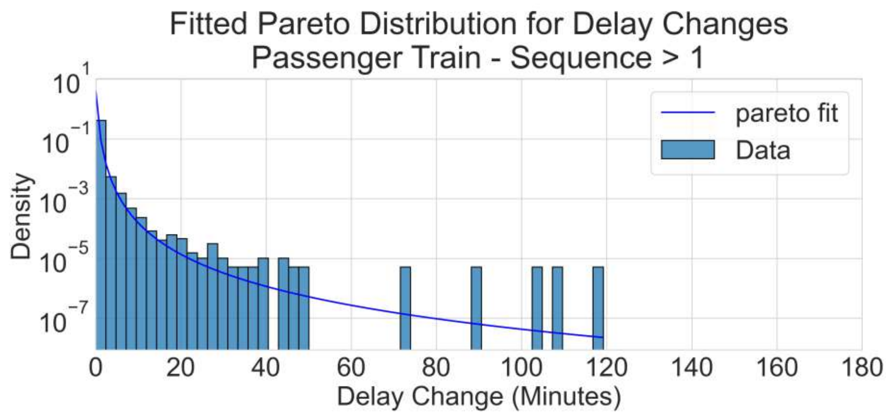


Figure 4.8: Pareto fit for Passenger Train Journey

Freight Train Start

The third scenario addresses the delays at the start of a freight train's journey. The Pareto distribution parameters and the goodness-of-fit statistics for this scenario are shown in the following table and plot.

Distribution	Pareto
Parameters	Shape: 6.4447, Location: -194.4331, Scale: 194.4497
D-statistic	0.040
n	597

Table 4.3: Pareto Distribution Fit for Freight Train Start Delay Changes

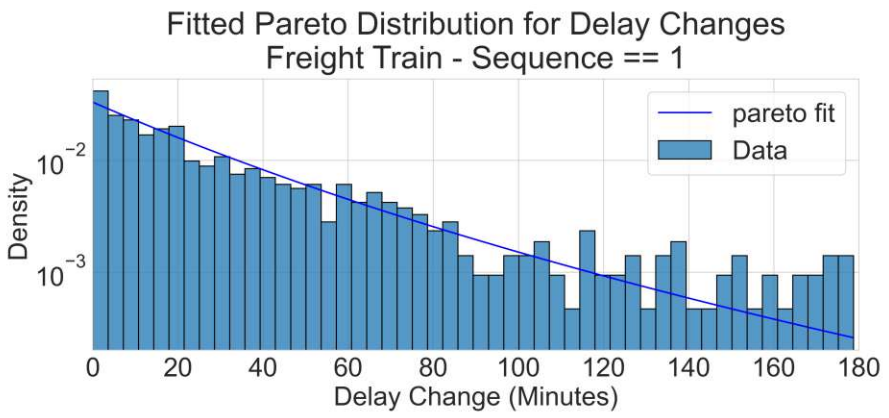


Figure 4.9: Pareto Fit for Freight Train Start

Freight Train Journey

The final scenario explores the delays occurring during the journey of freight trains. The following table and plot display the Pareto distribution parameters and the goodness-of-fit statistics.

Distribution	Pareto
Parameters	Shape: 1.0760, Location: -0.3725, Scale: 0.3892
D-statistic	0.045
n	16267

Table 4.4: Pareto Distribution fit for Freight Train Journey Delay Changes

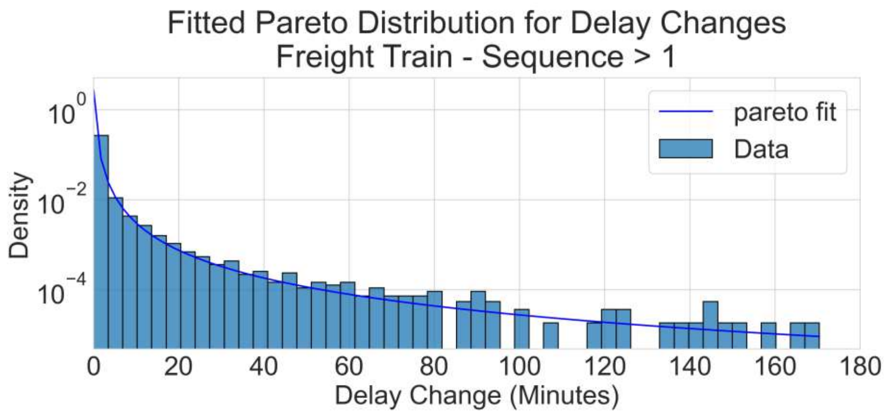


Figure 4.10: Pareto Fit for Freight Train Journey

With the distribution parameters for various scenarios established, the next step involved deriving upper boundaries for delay occurrence frequencies based on historical data. This analysis helped to set realistic rates for the agent-based model.

4.2.4 Upper Boundaries of Delay Frequencies Based on Historical Data

Based on the historical data, upper boundaries for the rates of occurrence of delays can be derived. This was done by assuming that each delay change is a PD. However, the exact values of these probabilities are not shown due to data confidentiality requirements.

These estimates were used as an upper boundary in the grid search described in Section 4.3.2, which aimed to find feasible PD occurrence rates.

4.2.5 Negative Delays and Negative Delay Changes

Negative delays also occur in the available data. A delay is negative if a train departs or arrives before its actual scheduled time. In real-world scenarios, this happens especially for freight trains, as these often depart hours before the scheduled time.

Negative delay changes happen if a train makes up time. The agent-based model cannot sample negative PD. In the model, a train can never depart or arrive before the scheduled time. The only way negative delay changes can be mimicked is the minimum duration. The minimum duration describes the minimal time it takes to travel from one OCP to the next or the minimum time a stop takes. The OEBB schedules their timetable so that trains must include a regular time buffer of at least 7% of the travel time [37]. This buffer was also implemented in the model for runs and stops. Consequently, late trains can make up time. However, enabling early departures from stations within the agent-based model is not possible.

4.3 Evaluation and Results of Synthetic Data Sampling

This section will elaborate on how the agent-based model was calibrated. The overall procedure involved the following steps:

1. **Sampled PD with Different Parameter Sets**: The agent-based model was calibrated by sampling PD using different sets of occurrence probabilities for the four estimated PD duration distributions.
2. **Compared Synthetic Data with Real-World Data**: The resulting synthetic data files were then compared with the real-world data using a defined comparison metric.
3. **Ranked Parameter Combinations**: Finally, the used parameter combinations were ranked based on the results from the comparison metric.

4.3.1 Distribution Comparison Method

To assess the suitability of the parameter setup, two distribution comparison metrics were utilized: the Kullback–Leibler divergence (KL-divergence) 4.3.1 and the K-S Test 4.2.1. It was aimed to achieve a more balanced evaluation ranking distribution based on these two metrics instead of a single one.

The delay change distributions for the four cases described in Section 4.2.3 were compared between the generated data and the historical data using these comparison metrics.

Definition 4.3.1 (Kullback-Leibler Divergence)**.** KL-divergence quantifies the deviation of one probability distribution from another reference distribution. For discrete probability distributions P and Q, the KL-divergence from Q to P is defined as:

$$KL(P \parallel Q) = \sum_i P(i) \log \left(\frac{P(i)}{Q(i)} \right) \tag{4.4}$$

where P and Q are probability distributions [38].

The following steps were taken to find feasible PD occurrence probability parameters:

1. **Defined Frequency Grid**: Defined a grid of feasible frequencies based on the estimated upper boundaries 4.2.4. With these probabilities, PD were sampled from the four different Pareto distributions 4.2.3.
2. **Ran Simulations**: Ran simulations for each parameter combination with two different seeds.
3. **Compared Distributions**: Compared the delay changes in the synthetic data to the historical data for:

- Passenger trains (sequence > 1)
- Passenger trains (sequence $= 1$)
- Freight trains (sequence > 1)
- Freight trains (sequence $= 1$)

4. **Normalized the Scores**: Normalized the KL-divergence and K-S Test results for the different distributions using min-max normalization to ensure comparability.

5. **Computed Weighted Mean**: Computed the weighted mean for both KL-divergence and K-S Test separately for each of the four cases. As the number of passenger trains is higher than the number of freight trains, freight and passenger trains were weighted differently. The ratio of passenger to freight trains was 5:1, but the weights were set to a ratio of 2:1 to consider the distribution of freight trains to a more noticeable degree. Additionally, sequence $= 1$ to sequence > 1 were weighted with a 2:1 ratio, founded on the fact that there are way more occurrences of trains with sequence > 1, but nevertheless, sequence $= 1$ is significant. This led to the weights of:

   - Passenger trains sequence $= 1$: $\frac{2}{9}$
   - Passenger trains sequence > 1: $\frac{4}{9}$
   - Freight trains sequence $= 1$: $\frac{1}{9}$
   - Freight trains sequence > 1: $\frac{2}{9}$

6. **Averaged Scores**: Averaged the scores from the same parameter setup but with different seeds.

7. **Ranked Scores**: Computed ranks for both KL-divergence and K-S Test statistics.

8. **Determined Final Ranking**: Ranked the parameter combinations based on the computed ranks for KL-divergence and K-S Test, producing a list of frequency parameters that most closely match the historical data.

Penalty Term

During exploratory calibration experiments, it was observed that although the delay change distributions were reasonably approximated, significant deviations from the historical data in the AGD occurred. To mitigate this issue, two penalty terms were introduced. Here, the KL-divergence and K-S Test between synthetic and historical data are computed for the AGD of passenger trains and freight trains. They were added to the before-described weighted means, with an overall weight of 1/3. The penalty term itself is again weighted with 2:1 for the passenger-to-freight ratio. This led to the following final weights:

- Passenger trains, sequence $= 1$: $\frac{2}{9} \times \frac{2}{3} = \frac{4}{27}$

- Passenger trains, sequence > 1: $\frac{4}{9} \times \frac{2}{3} = \frac{8}{27}$
- Freight trains, sequence $= 1$: $\frac{1}{9} \times \frac{2}{3} = \frac{2}{27}$
- Freight trains, sequence > 1: $\frac{2}{9} \times \frac{2}{3} = \frac{4}{27}$
- Penalty term 1 – AGD of passenger trains: $\frac{2}{3} \times \frac{1}{3} = \frac{6}{27}$
- Penalty term 2 – AGD of freight trains: $\frac{1}{3} \times \frac{1}{3} = \frac{3}{27}$

4.3.2 Grid Search for Optimal Parameters

A grid search was performed to find suitable parameters for calibrating the agent-based model, using the comparison method described and the estimated upper boundaries of occurrence probabilities from Section 4.2.4. The specific grid search parameters and final results are not disclosed due to data confidentiality.

The grid was run for two different seeds for each parameter combination. The results were ranked based on the comparison method described.

4.4 Plausibility Analysis

A plausibility analysis was conducted to compare several delay metrics between the synthetic and real-world data. This was done for a file of the best-ranked parameter combination in the before empirically determined ranking. The following subsections outline various plausibility checks. Before analyzing the data, the delay changes were filtered to a feasible time range for comparative analysis. Afterward, for all positive values it was normalized (min-max normalization 4.1.1) to ensure data confidentiality.

Delay Changes

The K-S Test test was used to evaluate the goodness-of-fit for the delay change duration distributions of the synthetic data compared to the historical data. The test was conducted for positive delay change durations. The results are summarized in Table 4.5 and shown in Figure 4.11.

Distribution	KS Statistic	$n_{\text{real-world}}$	$n_{\text{synthetic}}$
Passenger Trains sequence $= 1$	0.0724	3471	3172
Passenger Trains sequence > 1	0.0538	81773	48742
Freight Trains sequence $= 1$	0.0834	563	491
Freight Trains sequence > 1	0.0647	16252	14093

Table 4.5: K-S Test Results for Positive Delay Change Distributions Compared to Historical Data

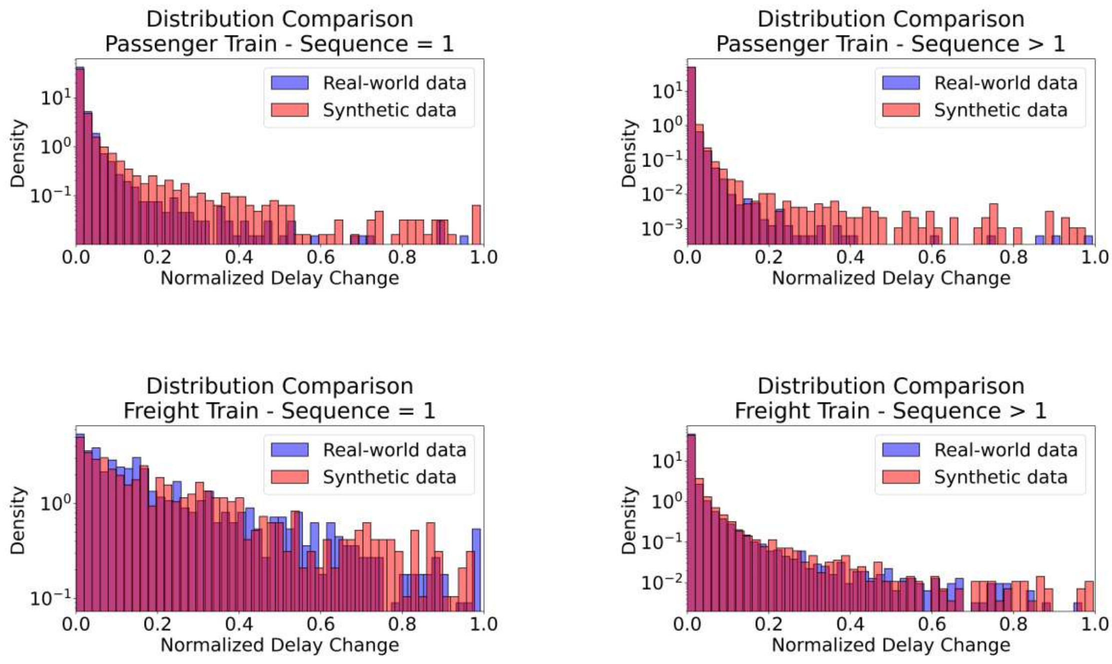


Figure 4.11: Distribution Comparison for Delay Changes of Passenger and Freight Trains

Negative Delay Changes

The K-S Test results for the negative delay changes, as shown in Table 4.6 and Figure 4.12, reveal significant deviations between the synthetic data and the real-world data. The higher K-S Test statistic values indicate that the distributions for these negative delay changes do not align as well with the historical data compared to the other scenarios.

Scenario	**KS Statistic**	$\mathbf{n_{real\text{-}world}}$	$\mathbf{n_{synthetic}}$
Passenger Trains	0.3739	117201	149504
Freight Trains	0.4563	25166	28074

Table 4.6: K-S Test Results for Negative Delay Change Duration Distributions (-10 to 0 Minutes) Compared to Historical Data

Delay Changes for Different Time Windows

Table 4.7 and Figures 4.13 present the K-S Test statistic results for passenger and freight trains, with a sequence number greater than one, for three different time windows (00:00-08:00, 08:00-16:00, and 16:00-24:00). The same minimum and maximum values for normalization have been used as in Section 4.4.

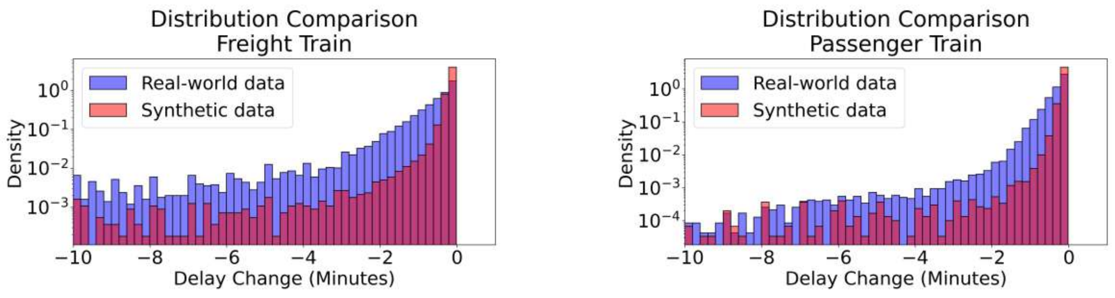


Figure 4.12: Comparison of Negative Delay Changes between Real-World and Synthetic Data

Scenario	KS Statistic	$n_{\text{real-world}}$	$n_{\text{synthetic}}$
Passenger Trains (00:00-08:00)	0.0375	19356	11491
Freight Trains (00:00-08:00)	0.0927	5733	5525
Passenger Trains (08:00-16:00)	0.0695	33665	20506
Freight Trains (08:00-16:00)	0.0413	5249	4315
Passenger Trains (16:00-24:00)	0.0709	32223	19917
Freight Trains (16:00-24:00)	0.0706	5833	4744

Table 4.7: K-S Test Results for Delay Change Distributions Compared to Historical Data across Different Time Windows

Overall Delays

The K-S Test results for a comparison of the AGD are shown in Table 4.8 and Figure 4.14.

Scenario	KS Statistic	$n_{\text{real-world}}$	$n_{\text{synthetic}}$
Passenger Trains	0.1038	166503	197040
Freight Trains	0.1600	28263	39300

Table 4.8: K-S Test Results for Departure Delay Distributions Compared to Historical Data

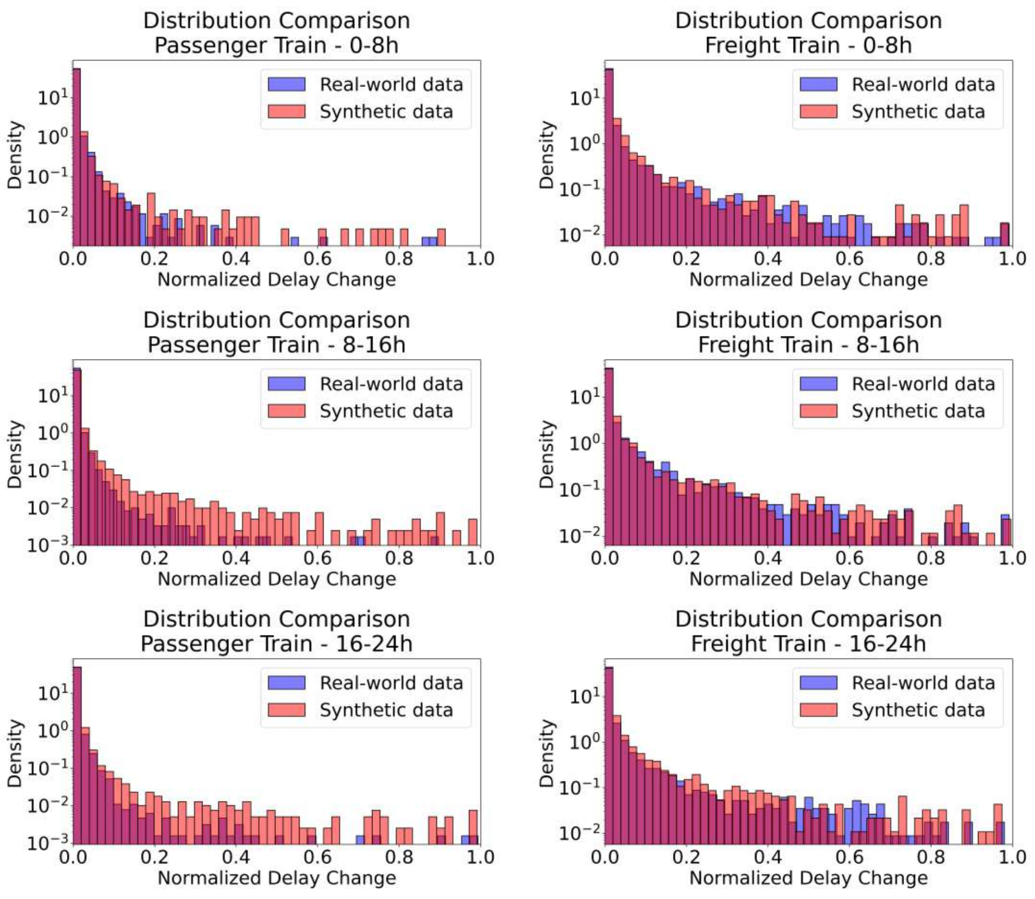


Figure 4.13: Distribution Comparisons for Delay Changes in Different Time Windows

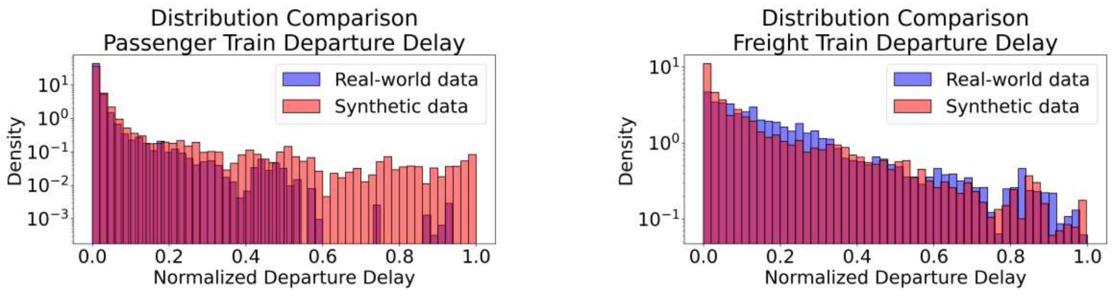


Figure 4.14: Departure Delay Distributions Compared to Historical Data.

4.4.1 Estimates based on Naive Model

To enhance the plausibility analysis, a boundary for the PD to SD ratio was estimated. This approximation was achieved by assuming that each delay change is a PD. The PD were fed into the agent-based model, which led to the results in Table 4.9.

Metric	**Percentage**
Percentage of nodes to classify (`delay_change` $\geq$ 60 seconds)	3.15
Percentage of nodes to classify where `primary_delay_label` is True (PD $\geq$ SD)	93.68
Percentage of instances where both `secondary_delay` $\geq$ 60 and `primary_delay` $\geq$ 60 relative to relevant nodes	1.85

Table 4.9: Estimates Based on Naive Model

Some valuable information for the plausibility analysis can be derived from these results:

Finding 1: The percentage of nodes to classify can be considered as an upper boundary, as this value would only be that high if each delay was a PD. Fewer cases of PD would generally result in fewer SD, leading to fewer nodes with delays $\geq$ 60 seconds.

Finding 2: The "Percentage of nodes to classify where `primary_delay_label` is True" is 93.68%, which is unexpectedly high. It can be expected that in a majority of cases, a smaller number of PD would likely even result in a higher number. Therefore, the estimated number can be considered as an approximate lower boundary.

Finding 3: For the "Percentage of instances where both `secondary_delay` $\geq$ 60 and `primary_delay` $\geq$ 60 relative to relevant nodes," it is expected that with fewer PD, this number would, in most cases, also decrease.

Important to note is that these estimated boundaries are not absolute. There exist scenarios where the relationships between PD and SD can be more complex. For instance, a reduction in PD occurrences might, counterintuitively, lead to an increase in SD in certain network configurations, which underscores the need for careful interpretation of these findings.

Comparison with Estimates

Table 4.10 compares the boundary estimates with the same metrics of the synthetic file. The synthetic file shows a higher percentage of nodes to classify, meaning it surpasses the upper boundary slightly. On the other hand, the lower boundary of the ratio of PD to SD is undershot by more than 10%. Although the lower boundary is merely an approximation, the values are still considerably different. Therefore, this discrepancy has to be considered as a weakness in the generated data.

Metric	**Naive Model Real-World Data (%)**	**Synthetic File (%)**
Percentage of nodes to classify (`delay_change` $\geq$ 60 seconds)	3.15	4.48
Percentage of nodes to classify where `primary_delay_label` is True (PD $\geq$ SD)	93.68	79.53
Percentage of instances where both `secondary_delay` $\geq$ 60 and `primary_delay` $\geq$ 60 relative to relevant nodes	1.85	1.62

Table 4.10: Comparison of Naive Model Estimates and Synthetic File as a Plausibility Check

Conclusion of the Plausibility Analysis

In conclusion, the plausibility analysis confirms that the generated synthetic data mirrors the real-world delay patterns observed within the Austrian rail network. However, it has to be pointed out that several weaknesses have been detected. Some of the analyzed distributions differ quite considerably from the real-world data regarding the K-S Test statistics. This especially holds true for the negative delay changes. Another weakness is the difference between the naive model estimates and the generated data 4.4.1. Nevertheless, the overall trends and key characteristics of delays are well-represented in the synthetic data, indicating that the chosen parameters for data generation are feasible.

Generation of Final Dataset

A final dataset was created based on the calibration experiments described in Section 4.3.2. It was decided that the top 10 parameter combinations would be used for the final dataset creation. However, due to confidentiality concerns, the specific combinations are not disclosed. Overall, 300 files were generated, with 30 files for each of the 10 combinations. Each file was created with a unique seed.

4.5 Limitations

The process of data sampling and generation in this study encounters several significant limitations that have to be pointed out:

1. **Limited Timespan of Historical Data**: The dataset used is constrained to a single day, limiting its diversity and robustness. Train delay patterns can vary significantly across different days, weeks, months, and seasons. Utilizing a larger and more diverse dataset would enable more thorough analysis and sampling, leading to more representative and accurate synthetic data.

2. **Early Departures**: The agent-based model cannot currently simulate that trains sometimes depart from a station before their scheduled time. In real-world scenarios, this regularly happens for freight trains at their first stop due to available capacities, which is represented in the recorded data as negative initial delays. This limitation reduces the realism of the model since these early departures are common. Future models should incorporate mechanisms to simulate these types of negative delay changes.

3. **Simplifying Assumptions**: Several simplifying assumptions were made during the data generation process, as described in Section 4.2.2. While these assumptions help simplify the sampling process, they do not fully capture the complexities of real-world scenarios. These assumptions were also set due to the limited data available. Future research should aim to refine these assumptions for more accurate data generation if more real-world data becomes available.

4. **Discrepancies to Estimates**: The differences between the naive model estimates and the generated data highlight some weaknesses in the synthetic data generation process. These differences indicate challenges of synthetic data in representing real-world scenarios.

CHAPTER 5

Data Preprocessing and Model Implementation

This chapter describes how the output data of the agent-based model is preprocessed before being transformed into a graph. Next, it elaborates based on what rules the individual nodes and edges are being created. Afterward, it gives an overview of how the GNN was implemented. For the detailed code of the data preprocessing, the graph creation, and the model implementation, please refer to the GitHub repository.[1]

5.1 Data Structure and Preprocessing

5.1.1 Structure of CSV

The output data of the agent-based model is stored in CSV files, with each row representing trains at different OCP for the times when the trains visit those checkpoints. Each row stores the features outlined in Table A.1. One file represents a whole run of the simulation, which equals approximately a full day of train traffic. To better understand the data structure, it is essential to understand the variable OCP, which represents checkpoints within the railway network where data is collected. These can include operational facilities such as stations, signal boxes, junctions, or crossings. If the feature OCP has the value *Pass*, it means that a train passed an OCP. *Stop*, on the other hand, means that the train did stop at the OCP. A delay change in a row with $OCP =$ Stop indicates that the change in delay happened at the stop itself, meaning the difference between the arrival and the departure at the station was longer than planned. Rows where $OCP =$ Pass indicate what happens on the sections between stations. If a delay change is shown in a row with $OCP =$ Pass, it means that it occurred on the section between the OCP of the row and the OCP of the following sequence of this train.

[1] https://github.com/maximilianvie/GNN-Based-Train-Delay-Disaggregation

5.1.2 Target Variable

The target variable in this study is the PD, more precisely, delay changes caused by external influences on the train network. Delay changes can be as short as only a few seconds or as long as several hours. In this study, delay changes from 0 to 59 seconds are considered noise as they are not particularly relevant for model training. Therefore, predictions should only be made on rows with at least 60 seconds of delay change. In the real-world data, this only applies to around 3% of the rows (see Section 4.4.1). This design choice was made to avoid learning from insignificant delay changes. It enables faster and more robust model training.

Two types of GNN are trained in this study: one which has a regression task and a second model that only classifies delay changes, as described below:

- **Regression Task** - For the regression task, the goal of the GNN is to estimate for each row to what extent a delay change is a PD. The difference between the delay change and the PD then consequently makes the SD.
- **Classification Task** - For the classification task, each row gets a label PD or SD. This is based on the rationale that for approximately 98% (estimated in Section 4.4.1) of rows, either PD or SD exist, which is why in most cases, it is sufficient to classify a delay change as only PD or SD.

5.1.3 Handling Missing Values

Overall, 300 CSV files were used as a final dataset (Section 4.4.1). Each synthetic file had 384 missing values in the longitude and latitude columns. These were filled using a forward-filling technique for the individual trains, which means that the coordinates of the previously visited station were used as the value. For missing departure times, the corresponding arrival time at the same station was used, and vice versa.

5.2 Feature Selection

The output data files of the agent-based model contain 29 features, as shown in Table A.1 in Appendix A.2. Within a feature selection step, this number has been reduced to six essential input features for the final GNN. This section will briefly outline how the attributes have been selected.

5.2.1 Feature Selection Rationale

Feature selection in deep learning is a non-trivial task due to the potential presence of non-linear relationships between individual features and the target variable. The approach to feature selection can be quantitative (e.g., correlation analysis) or qualitative (e.g., domain knowledge).

In this study, the features were first filtered based on expert opinion, representing a qualitative approach. Afterwards, in the experimental stage, the features were further reduced by a non-exhaustive exploratory feature selection. This was done by testing how different input feature combinations affected the prediction accuracy in a dataset created for exploratory experiments. This simplified approach was chosen for several reasons:

- This work aims to develop a Proof of Concept (PoC), which is why the focus is on finding a functional, but not necessarily optimal, solution.
- The primary task of the GNN is to learn the different delay propagation mechanisms within the graph. It is hypothesized that these mechanisms can be learned to a large degree from the graph structure itself.
- Each additional feature increases the risk that the model overfits the synthetic data. As the final model will be used to run inference on real-world data, a strong focus has to be on generalizability.

The selection of features for exclusion and inclusion before preprocessing was guided by several criteria, including:

- Relevance to the predictive task
- Redundancy

Table A.2 in the Appendix A.2 summarizes the features that were excluded, along with the rationale for their exclusion.

5.2.2 Final Feature Selection

In this subsection, the final feature set of the model is presented. The original features (Appendix A.1) were filtered based on a set of criteria. This filtered set is shown in Appendix A.3. The final set of features (see Table 5.1) was then determined based on exploratory experiments that evaluated the effects of different feature combinations on the accuracy within the training dataset. Additionally, the attributes `is_supernode` and `train_unique_sequence` were added.

5.3 Normalization and Feature Encoding

5.3.1 Encoding Cyclical Time Features

Time features should be transformed before being used as input features since a model might otherwise perceive values such as 23:00 and 00:00 as being very far apart, although they are very close. To mitigate this issue, the periodic nature of date time was leveraged. The daytime values were transformed using sine and cosine functions to preserve their

Retained Feature	**Description**	**Data Type**
`passenger`	A binary indicator of whether the train is a passenger (true) or freight train (false).	bool
`ocp_type`	Indicates if a train stops (true) or passes (false) an OCP.	bool
`arrival_delay_in_seconds`	Indicates the train's arrival delay at a checkpoint.	int
`departure_delay_in_seconds`	Indicates the train's departure delay at a checkpoint.	int
`train_unique_sequence`	Represents the sequence number that shows the train's current position in its journey.	int
`is_supernode`	Indicates if a node is a supernode.	bool

Table 5.1: Input Features for GNN

cyclical characteristics. Each time component, like second, minute, hour, or day, is mapped onto a circle in this transformation approach. This approach maintains continuity since the end of the cycle is adjacent to its start.

The transformation process has two steps for each cyclical component of datetime values:

Extraction of Time Components: From each datetime value, the time component of interest is extracted. This component will be used for the next step.

Application of Sinusoidal Functions: Next, both sine and cosine functions are applied to the extracted time components using these formulas:

- Sine Transformation: $\sin(\frac{2\pi}{T} \times \text{Component Value})$
- Cosine Transformation: $\cos(\frac{2\pi}{T} \times \text{Component Value})$

Here, T is the maximum value of the cyclical component (for example, 24 for the hours of a full day). This step computes the corresponding coordinates and projects each time point onto a unit circle.

5.3.2 Normalization of Continuous Features

The continuous numerical features in this study, such as the arrival delay in seconds, were normalized using *z-score normalization*. This method adjusts the features and labels to achieve a mean of 0 and a standard deviation of 1. As a consequence, it is ensured that all features contribute equally to the model's learning process since they are scaled the same way. The normalization of the features also helps gradient descent-based optimization methods converge faster, making the training of ANN more efficient.

The standardization is applied to each feature and label independently. It is described by the formula in Definition 5.3.1.

Definition 5.3.1 (Z-Score Normalization)**.**

$$x_{standardized} = \frac{x - \mu}{\sigma} \tag{5.1}$$

where x is the original value of the feature or label, μ is the mean, and σ is the standard deviation.

The standardization parameters (mean and standard deviation) are computed only on the training data and then applied to all dataset splits to prevent information leakage.

It is essential to mention that, as elaborated in Section 5.1.2, within the model training, the loss will only be computed on nodes with a delay change of at least 60 seconds. In the code implementation, these nodes are described as `nodes_to_train_on`. Only around 3% of the nodes of a graph in this study fall into this category. To compute the normalization parameters for the target feature, the PD, only the corresponding values of the `nodes_to_train_on` were used. This ensures that the standardization only focuses on relevant delays.

5.3.3 Encoding of Categorical Features

Categorical node features are not straightforward to feed into an ANN. While they could be represented as integer values, this approach is not recommended because the network might incorrectly interpret the integers to have a continuous relationship. For that reason, these features were handled using embedding layers. Through this process, each category is mapped to a vector embedding, which is updated during model training, allowing the GNN to learn the relationships between different categories.

5.4 Graph Structure

The next goal is to transform the preprocessed CSV data into a graph structure that enables delay information to be efficiently propagated through the network by the message-passing mechanism of the GNN. This section explains how tabular data is mapped into a graph, specifically how nodes and edges are created based on predefined rules.

5.4.1 Creation of Nodes and Supernodes

In the defined graph structure, each node represents a train visiting an OCP at a specific time and can be uniquely identified by the combination of the train number and the train sequence number. Each node stores values for the features defined in Section 5.2.2.

Inspired by the concept of hierarchical message-passing introduced in [23], this study introduces a specific type of node referred to as a *supernode*. Each train is associated with a single supernode, which connects to all its corresponding train nodes, enabling long-range message passing. Additionally, supernodes can connect to one another as shown in Figure 5.1.

They carry the same features as the other nodes, but their value is determined by the corresponding single checkpoint nodes of the same train. Supernodes are created with attributes that are either the maximum or mode from the characteristics of the train's nodes.

5.4.2 Edge Creation

The graph's edges illustrate the sequence of train movements across the network and the interactions among trains at common stations. In contrast to nodes, they do not carry any features. During an exploratory stage, experiments have shown that bidirectional edges led to higher performance than non-bidirectional graphs, which led to the decision to add edges in both directions for each pair of nodes that should be connected. Three types of edges were created:

- **Sequential Edges:** These edges connect consecutive checkpoints for each train. An edge is created between these nodes if a train moves from one node to another.

- **Temporal Proximity Edges:** If two trains pass through the same station within a specified time window (defined as 180 seconds), edges are created between their respective nodes. This type of edge allows potential delays to propagate through the network.

- **Supernode Edges:** Each supernode is connected to all of the train's nodes. Additionally, supernodes are connected to each other if, between the two trains, a temporal proximity edge is created.

In Figure 5.1, the graph structure of a single train is shown. The node defined as *Train_A* represents the supernode. A node was created for each unique OCP of a train's journey. In the node naming, *Pass* and *Stop* represent the OCP type, while the number represents the sequence number of the train at the corresponding checkpoint.

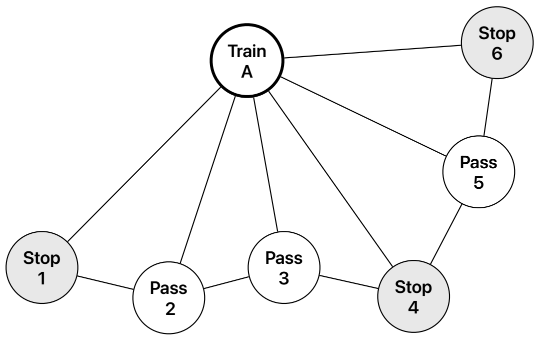


Figure 5.1: Graph Structure of a Single Train

Figure 5.2 shows a scenario where two different trains happen to visit the same station within a 180-second time window. Therefore, node *Stop_4_Train_A* and node *Stop_1_Train_B*, which both represent the described station, are connected by an edge. Consequently, as a result of the edge rules, the two supernodes are also connected. In this scenario, the utility of the supernodes is apparent. The minimum distance from *Stop_1_Train_A* to *Stop_7_Train_B* is 3. Without the implementation of supernodes, the distance between those two nodes would be 10. As a PD in the first node can have an influence on the delay in the second node, it is important that messages can be passed efficiently between the two nodes. Not using supernodes would, therefore, require the use of many more message-passing layers to achieve the same travel distance.

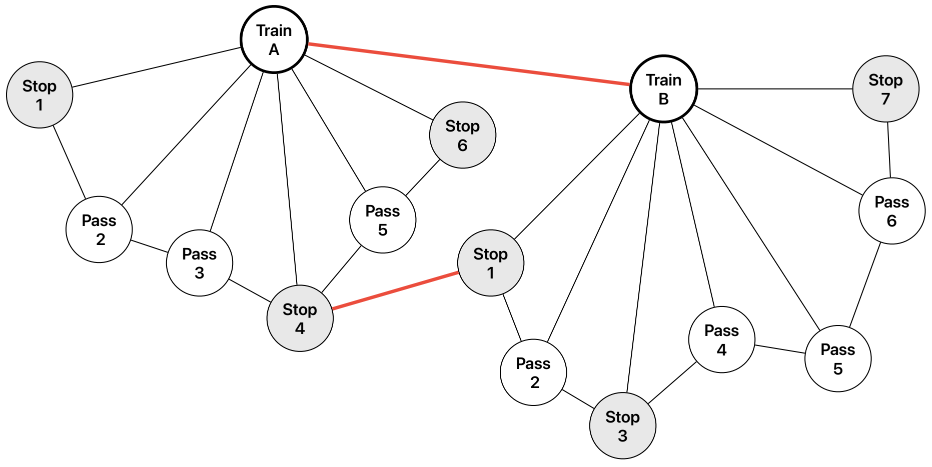


Figure 5.2: Graph Structure of Two Trains Visiting the Same Station in a Certain Timeframe

The transformation of the CSV files to the explained graph structure usually led to one large graph with more than 200,000 nodes and multiple smaller graphs with less than 1000 nodes. All graphs with less than 1000 nodes were considered negligible and discarded for model training.

5.5 Graph Neural Network Implementation

5.5.1 GraphGym Framework

In this thesis, the GNN was implemented using the GraphGym [39] framework, which is built on top of PyTorch Geometric [40]. GraphGym is a platform that was designed to enable scalable and reproducible experimentation with different GNN architectures. Its modular design allows efficient testing of how different model configurations (e.g., layer numbers or activation functions) influence the target metrics. Additionally, GraphGym's standardized evaluation pipeline ensures consistent and reproducible results across different experiments.

The specific implementation details and configuration used for this GNN can be found in the accompanying repository.

5.5.2 Key Facts about the GNN

This section provides an overview of the key aspects and methodologies employed in the GNN used in this study. For detailed code implementation, please refer to the GitHub repository.[2]

The default configuration of the GNN is available in the Appendix A.3.1. Below, essential information about the training is outlined:

- **Data Splitting:** Overall, 80% of the graphs were used for training, 10% for validation, and 10% for testing. This split was done by random selection and was the same for each experiment. The learning rate was dependent on the validation results.
- **Dataset Statistics:**
  - Number of graphs in the final dataset: 300
  - Average number of nodes per graph: 279,730
  - Average number of edges per graph: 1,418,394
- **Loss Functions:**
  - Regression Task: L1 loss
  - Classification Task: Cross-entropy loss
- **Activation Function:** ReLU (Rectified Linear Unit)

[2]https://github.com/maximilianvie/GNN-Based-Train-Delay-Disaggregation

- **Batch Size:** A batch size of one was used due to the large size of the graphs. Large graphs contain a significant amount of diverse data. When training on them, independent loss values are computed at each node due to the mechanisms of GNN. Therefore, a batch size of one in this context can capture the patterns and variability similar to a larger batch size in other domains.
- **Optimizer:** Adam (Adaptive Moment Estimation) was used as an optimizer. The learning rate was adjusted based on the performance on the validation dataset and was reduced when the target metric no longer showed improvement.

CHAPTER 6

Results

This chapter aims to outline the results obtained from the final experiments. The results are compared to a naive model to establish a baseline. The chapter is divided into sections detailing the evaluation of the GNN model performance on a classification task, a regression task, and the application to real-world data.

6.1 Evaluation of GNN Model Performance on Synthetic Data

In this section, the model training and evaluation process for the synthetic data is described. Next, the exact results of the experiments are outlined. The models are compared to a naive model in which every delay is assumed to be a PD. Two models are trained, a classification and a regression model:

Classification Model

In the classification model, all nodes within the graph that have a delay change of at least 60 seconds are classified as either PD or SD. The decision to train a classification model is based on the estimation that only around 1.85% of nodes exhibit both PD and SD of at least 60 seconds, as outlined in Section 4.4.1. If both types of delays are at least 60 seconds, the label is PD if PD >= SD. All delay changes lower than 60 seconds are considered insignificant and are treated as PD. This drastically reduces the number of nodes to predict and allows the model to learn primarily from nodes with high significance.

Regression Model

The regression model aims to predict for each delay change of at least 60 seconds the exact amount of PD. So if a delay change of 1000 seconds consists of 800 seconds PD and 200 seconds SD, it should predict 800. This model, therefore, follows the actual goal of the thesis of creating a disaggregation model. Nevertheless, it must be pointed out that in most cases, PD and SD most likely appear isolated. In the regression model, all delay changes lower than 60 seconds are considered insignificant and treated as PD to enable the model to train on high-delay nodes.

6.1.1 Hyperparameter Tuning

Hyperparameter tuning was conducted to find a good set of parameters for the GNN. As the design space in GNN is very large and this work aims to provide a PoC, the search space was limited to two parameters considered essential for the network's performance. On a reduced dataset of 100 files, several exploratory hyperparameter tunings have been carried out to estimate a feasible range of parameters. A final tuning was carried out for the full dataset. This study only lists the results of this final tuning.

Similarly, several feature sets have been investigated in an exploratory phase. In the thesis, only the results for the final feature set (see Table 5.1) are listed. Here, a reduced set of features was chosen because, during the exploratory phase, overfitting occurred with several attributes. This set of features might have the benefit of capturing only the most essential information of the graph, and therefore, it might generalize better to real-world data.

6.1.2 Classification Task

The following hyperparameter tuning grid (see Table 6.1) was defined for the classification task, of which the values are based on exploratory experiments. The remaining parameters were kept at default values as mentioned in the Appendix A.3.1.

Hyperparameter	**Search Space**	**Rationale**
Number of Layers	[3, 4, 5, 6]	The number of layers is highly important since it determines how far messages can be passed.
Inner Dimension	[60, 80, 100]	The inner dimension defines each layer's feature space size and consequently directly influences the number of parameters.

Table 6.1: Hyperparameter Tuning Grid Search

The exact results of the hyperparameter tuning can be seen in Appendix A.3.2. To demonstrate the stability of the GNN training, several learning curves are available in the Appendix A.3.2. For all further experiments, the learning curves showed similar patterns and stability and were therefore omitted.

The best-performing parameters for the classification task based on the accuracy of the grid search are:

- Number of Layers: 6
- Inner Dimension: 100

Based on these parameters, a final model was trained. The results of the model on the test set are compared to the naive model in Table 6.2.

Metric	**Best Performing Model**	**Naive Model**
Epochs	49	-
Number of Parameters	316601	-
Accuracy	0.9602	0.7820
Precision	0.9732	0.7820
Recall	0.9768	1.0000
F1 Score	0.9750	0.8773
AUC	0.9890	0.5000

Table 6.2: Test Set Results for Best Performing Classification Model and Naive Model

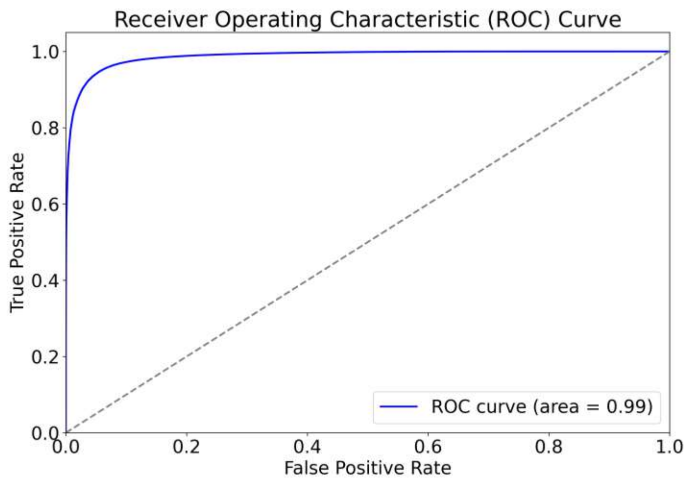


Figure 6.1: Receiver Operating Characteristic (ROC) Curve for Trained Model

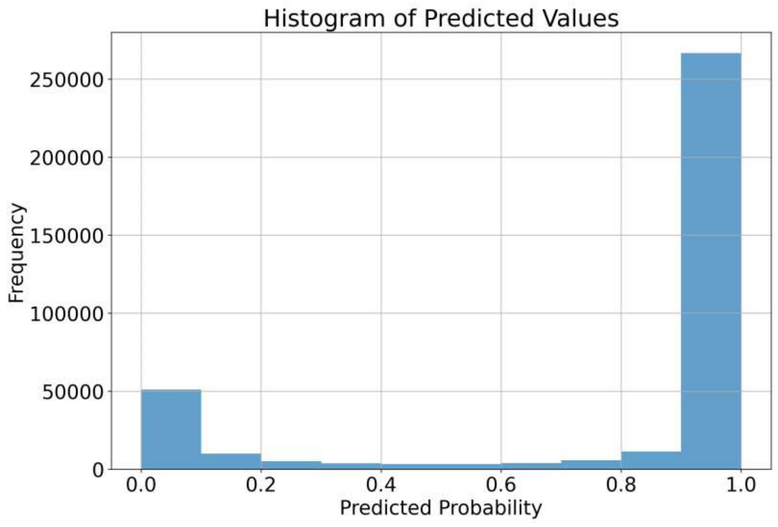


Figure 6.2: Histogram of Predicted Values

Table 6.2 shows that the classification model significantly outperforms the naive model across all key metrics. With an accuracy of 0.9602 and a F1 Score of 0.9750, the GNN can be considered as highly reliable. This is also underscored by the Area Under the Curve (AUC) value of 0.9890, which shows that the model is very effective in distinguishing between PD and SD. Figure 6.2 displays a histogram for the prediction probabilities of the classification model, which indicates that in a majority of cases, the model is confident in its predictions and assigns probabilities close to 0.0 and 1.0.

6.1.3 Regression Task

A hyperparameter tuning grid was determined for the regression task based on exploratory experiments. The grid is depicted in Table 6.3.

Hyperparameter	**Search Space**
Number of Layers	[5, 7, 9]
Inner Dimension	[40, 55, 70]

Table 6.3: Hyperparameter Tuning Grid Search - Regression Task

Exact metrics of the hyperparameter tuning are available in the Appendix A.3.3. The best-performing parameters for the regression task based on the grid search are:

- Number of Layers: 5
- Inner Dimension: 40

Table 6.4 shows the results of the regression model on the test set. The target variable was z-normalized to optimize the model training.

Metric	**Best Performing Model**	**Naive Model**
Epochs	49	-
Number of Parameters	43881	-
Mean Absolute Error (MAE)	0.0450	0.3414
Mean Squared Error (MSE)	0.1478	3.5323
R-Squared (R2)	0.8558	-2.5323
Spearman's Rank Correlation	0.8748	0.5068
Root Mean Squared Error (RMSE)	0.3845	1.8799

Table 6.4: Test Set Results for Best Performing Regression Model and Naive Model

For the z-normalized target variable, the model GNN achieved a Mean Absolute Error (MAE) of 0.0450 and a Mean Squared Error (MSE) of 0.1478. These values indicate that, on average, the predictions of the model deviate by 0.0450 standard deviations from the actual values. To put the results in relation, it can be compared to the baseline model, which shows higher errors with a MAE of 0.3414 and a MSE of 3.5323.

The R-squared (R2) value of 0.8558 shows that the GNN explains 85.58% of the variance in the outcome variable. Another metric to look at is Spearman's rank correlation, which has a value of 0.8748, reflecting a robust monotonic relationship between predicted and actual values. Concurrent with the results of the classification model, it is shown that the model is capable of learning the difference between PD and SD, predicting the target variable with high accuracy and outperforming the naive model.

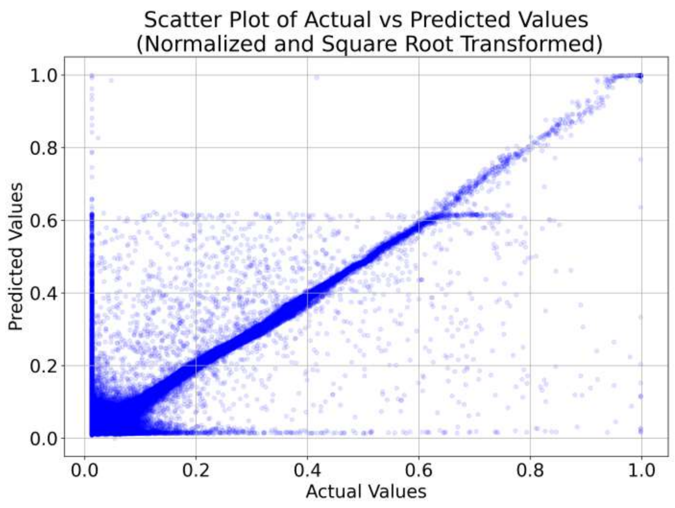


Figure 6.3: Scatter Plot of Actual Values vs Predicted Values (Normalized and Square Root Transformed)

The predictions for the full test set are plotted against the actual values in a scatter plot (Figure 6.3) to verify the plausibility of the results. The plot shows a clear linear trend along the diagonal, which is what is to be expected in a good regression model. Due to the concentration of many data points in the 0 to 0.1 range after min-max normalization, a square root transformation was applied to better spread these values and enhance the visibility of the trend. Despite some outliers and deviations, the plot complies with the results in Table 6.4.

6.2 Application to Real-World Data

In this section, the two trained models are applied to real-world data and compared to the naive model.

6.2.1 Analysis Using the Agent-Based Model

The final aim is to apply the trained models to the historical data and run inference. Here, the problem arises that no ground truth is available for the real-world data. Therefore, evaluating whether the models make valid predictions is not straightforward. For this reason, the agent-based simulation model is leveraged to validate the trained models.

If it is assumed that the GNN and the agent-based simulation model work without error, it would be possible to disaggregate real-world AGD using the GNN, feed the resulting PD back into the agent-based simulation model, and obtain the same AGD as a result.

The mechanisms of this loop are leveraged and used for the validation of the GNN, as displayed in Figure 6.4. It is congruent with the Problem Definition in Section 1.2.2 and the computed error in Equation 1.3.

Due to the multi-step nature of this validation method, several sources of error, particularly the inaccuracies inherent to the agent-based model, must be considered, as they distort the final validation metrics. These limitations are pointed out in Section 6.2.1.

Classification Model

In the classification model, all delay changes up to 59 seconds are considered as PD. For all delay changes of 60 seconds or higher, the GNN classifies if a delay is a PD or SD.

A delay change is taken as PD input to the agent-based model if it is classified as PD by the GNN. Two different softmax thresholds (0.1 and 0.5) are used to evaluate the classification model. The softmax values are the output of the classification model and represent the probability that a node belongs to a specific class, with 1 representing PD and 0 representing SD.

By adjusting the softmax thresholds, it can be analyzed how different threshold values impact the classification performance.

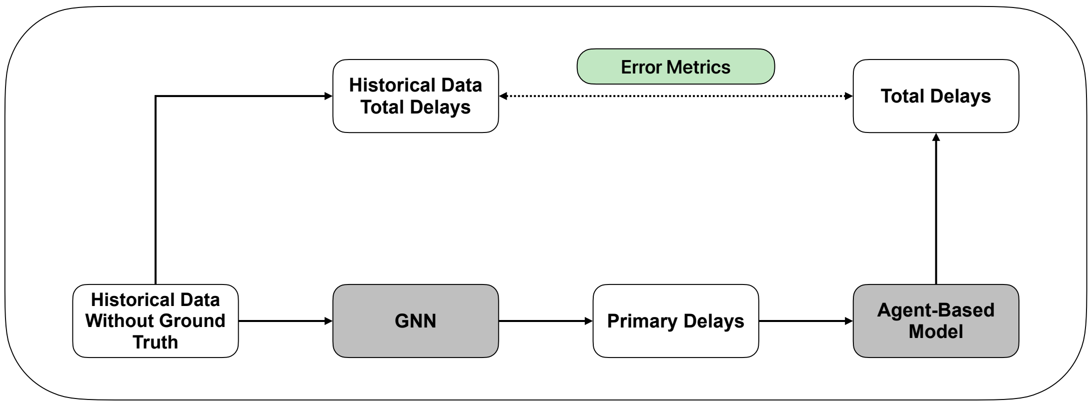

Error Metrics
Historical Data Total Delays
Total Delays
Historical Data Without Ground Truth
GNN
Primary Delays
Agent-Based Model

Figure 6.4: Simulation-Based Validation of the GNN on Real-World Data

Results

A new performance metric, the Weighted Absolute Percentage Error (WAPE) (Definition 6.2.1), is used to ensure data confidentiality. This metric is extended by splitting it into overestimation and underestimation components (Definition 6.2.2) to better understand whether WAPE arises from false positives or false negatives in the classification model or from excessively high or low PD predictions in the regression model.

Definition 6.2.1 (Weighted Absolute Percentage Error (WAPE))**.** The WAPE measures the overall prediction accuracy by calculating the sum of absolute errors as a percentage of the sum of actual values:

$$\text{WAPE} = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{\sum_{i=1}^{n} |y_i|}$$

where y_i is the actual value, $\hat{y}_i$ is the predicted value, and n is the total number of observations.

Definition 6.2.2 (WAPE Overestimation and Underestimation)**.** The WAPE Overestimation measures how much the model overpredicts relative to the actual values. The WAPE Underestimation, in contrast, quantifies the relative magnitude of underestimation errors compared to the actual values.

If the index sets

$$I^+ = \{i \in \{1, \dots, n\} : y_i < \hat{y}_i\}, \text{ and } I^- = \{i \in \{1, \dots, n\} : y_i > \hat{y}_i\}$$

are defined, then

$$\text{WAPE Overestimation} = \frac{\sum_{i \in I^+} |y_i - \hat{y}_i|}{\sum_{i=1}^{n} |y_i|}$$

and

$$\text{WAPE Underestimation} = \frac{\sum_{i \in I^-} |y_i - \hat{y}_i|}{\sum_{i=1}^{n} |y_i|}.$$

Table 6.5 shows a comparison of the different models applied to the real-world data.

Model	**Metric**	**Value**
Classification Model (Threshold 0.1)	R2	0.70
	WAPE	43.95%
	WAPE Overestimation	22.92%
	WAPE Underestimation	21.02%
Classification Model (Threshold 0.5)	R2	0.57
	WAPE	50.35%
	WAPE Overestimation	18.80%
	WAPE Underestimation	31.55%
Regression Model	R2	0.44
	WAPE	58.66%
	WAPE Overestimation	17.64%
	WAPE Underestimation	41.01%
Naive Model	R2	0.86
	WAPE	33.17%
	WAPE Overestimation	30.19%
	WAPE Underestimation	2.99%

Table 6.5: Comparison of Models on Real-World Data

As seen in Table 6.5, the naive model outperforms all other models on most metrics. However, it is inferior to the other models in terms of WAPE Overestimation.

The classification models outperform the regression model in all metrics except WAPE Overestimation. Among them, the model with a softmax threshold of 0.1 performs better regarding the R2 and overall WAPE.

Limitations of the Simulation-Based Validation

Several limitations arise when using this validation approach due to its multiple steps. The first limitation is that errors within the agent-based model also influence the final result, distorting the evaluation metrics.

The second limitation is the occurrence of model-based error distortions. In the simulation-based validation method, errors in the predictions of the GNN are not weighted equally in the final evaluation metric. Misclassifications influence the delays of other trains differently, depending on factors such as time and location. For example, a delay misclassified as a SD by the classification model would not be sampled in the simulation-based validation due to its classification. Consequently, all knock-on delays that would occur if it were correctly classified will not exist. As the error metrics are computed for the AGD, the absence of these resulting knock-on delays would further increase the error. This results in high-impact misclassifications having a much greater influence on the error metrics than low-impact misclassifications. A high-impact misclassification could be a sudden 30-minute delay during rush hour on a high-frequency route. In contrast, a low-impact misclassification could be a 30-minute delay at night. While these two misclassifications would be weighted equally in a classical ground truth validation, the weight might differ significantly in the simulation-based validation.

CHAPTER 7

Discussion

The results chapter showed that the GNN perform well on the synthetic data but struggle with generalizing to the real-world data. This discussion aims to find the underlying reasons for that discrepancy. The goal is to explore why the models have a drop in performance when being applied to the historical data and what improvements can be implemented to enhance their ability to generalize on real-world scenarios.

First, the results obtained from the synthetic data will be summarized and evaluated. Following this, an in-depth analysis of the model's results on real-world data will be conducted. The limitations of the research will also be addressed, which include data constraints, methodological assumptions, and model-specific challenges.

Additionally, the implications of the findings for the railway sector will be examined with a focus on how GNN can be utilized to address challenges within train networks. Finally, several directions for future research will be proposed.

7.1 Result Analysis

7.1.1 Summary of GNN Results for Synthetic Data

Chapter 6 provided results on the experiments on the synthetic data and the real-world data using the simulation-based validation approach.

The results on the synthetic data (Section 6.1) can be considered very promising, as they show that the overall proposed architecture within this work is capable of learning delay propagation mechanisms within train networks. They also point out that these networks can disaggregate AGD into PD and SD efficiently. This is a valuable finding as no literature was found where GNN were used to tackle this problem. It was shown that the task can successfully be formulated and described as a regression and a classification problem.

The classification score of 96% accuracy can be considered very accurate. The high precision and recall values show that the model is reliable and that the model captures a majority of positive cases and classifies them correctly.

The results of the regression model also show promising outcomes. The z-normalized MAE of 0.0450 indicates that, on average, the model's predictions are only 0.0450 standard deviations from the actual values. This can be interpreted as accurate, especially when relating it to the inferior naive model. The effectiveness of the model is also apparent by the R2 value, which shows that 85.58% of the variance in the target variable is explained by the model.

The models outperform the naive model in regression and classification scenarios.

Based on the results of the synthetic data, it can be concluded that the proposed architecture is capable of learning delay propagation patterns within train networks. This implies that GNN is a suitable method for a range of delay-related tasks.

7.1.2 Analysis of GNN Results for Real-World Data

Within this section, the results of the GNN experiments on real-world data are analyzed.

The experiments (Section 6.2) show, that the naive model outperforms all GNN when applied to real-world data. The naive model performs surprisingly well on the real-world data. This indicates a very high amount of PD in the ground truth, which would be in line with the PD approximation in Section 4.4.1.

Analysis of Classification Model Results

To get a better understanding of the classification model, the softmax predictions are analyzed.

In Figure 7.1, it can be observed that the model is, for most values, confident regarding its predictions, meaning having softmax values close to 0 or 1.

More insight can be gained by varying the classification threshold for the softmax values, which influences the PD to SD ratio. Table 7.1 shows for three different threshold values the resulting PD to SD ratios. The overall ratio is around 80:20, which is precisely the ratio in the synthetic training data. This strongly indicates that the GNN overfit to the synthetic data, particularly when considering the ratio estimation (Section 4.4.1) suggesting a potential PD share of around 94% in the ground truth - a figure that, while informative, should be interpreted cautiously.

The table also shows that the share of PD is higher for freight trains compared to passenger trains. It can be expected that, in reality, it is the opposite since freight trains have a lower operational priority compared to passenger trains and, therefore, might experience more frequently SD.

Next, it is analyzed how the predicted PD and SD are distributed across sequence numbers. Before plotting the data, the results were filtered by threshold values. All

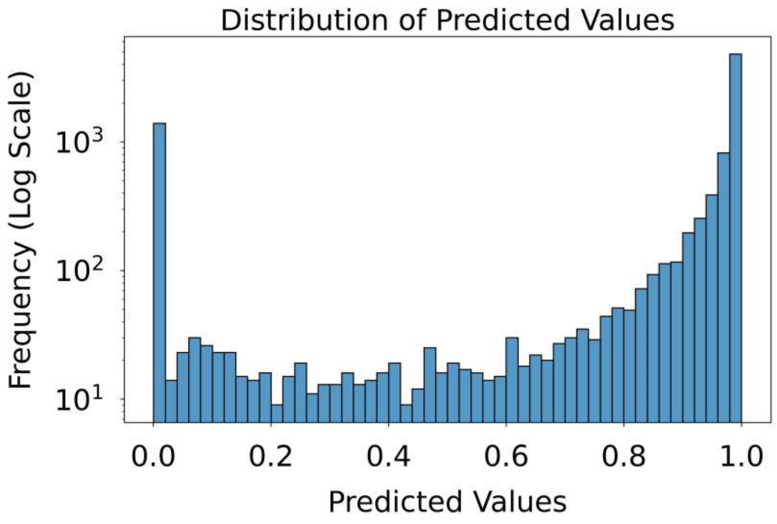


Figure 7.1: Distribution of Predicted Values (All Trains, Log Scale)

Softmax Threshold	All Trains		Freight Trains		Passenger Trains	
	Primary Delay	Secondary Delay	Primary Delay	Secondary Delay	Primary Delay	Secondary Delay
0.1	0.84	0.16	0.97	0.03	0.75	0.25
0.5	0.80	0.20	0.95	0.05	0.71	0.29
0.9	0.71	0.29	0.81	0.19	0.65	0.35

Table 7.1: Primary and Secondary Delay Ratios at Different Thresholds

values lower than 0.05 count as SD, and all values higher than 0.95 as PD. The filtered datasets were then plotted as histograms with the sequence numbers on the x-axis. This type of plot explains which sequence number the model usually classifies a delay change as either PD or SD. Figure 7.2 displays the described histogram for all threshold values lower than 0.05, so the defined SD. Although the y-axis is log-scaled, it can be seen that SD primarily occurs at the first sequence.

Figure 7.3 shows the same type of histogram for threshold values higher than 0.95, so PD. Here, the pattern is different. The values are more equally distributed but also show a peak at the beginning. The start peak might also be caused by overall more frequent delay changes at the start and, therefore, more nodes to classify. The observed downward trend with rising sequence numbers can be explained by the fact that some train trips are short and, therefore, only have a small number of sequence numbers.

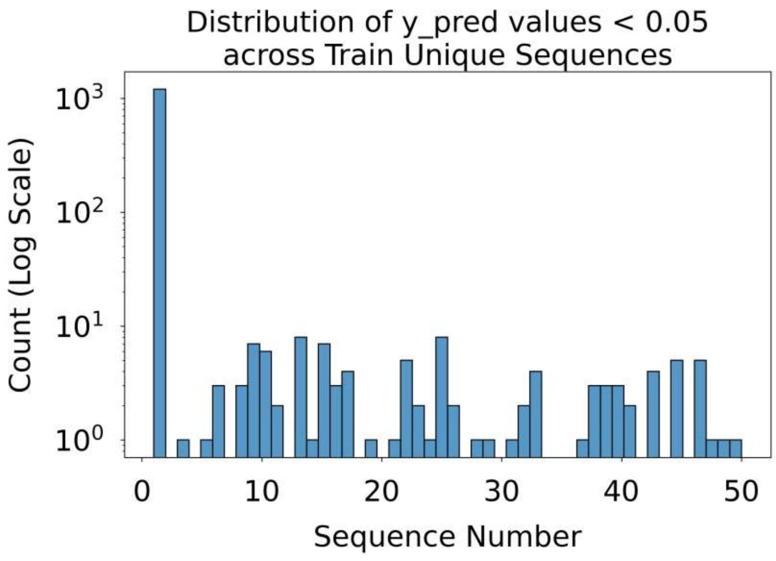


Figure 7.2: Distribution of y_{pred} Values < 0.05 Across Sequence Numbers

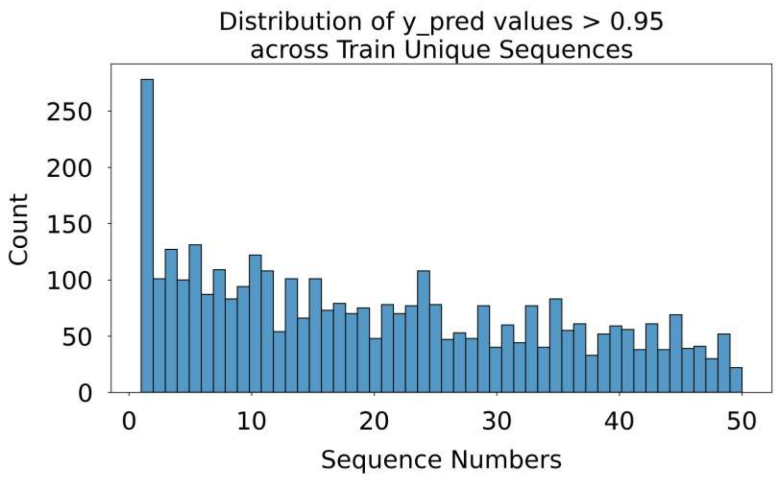


Figure 7.3: Distribution of y_{pred} Values > 0.95 Across Sequence Numbers

Analysis of Regression Model Results

In this section, the results of the regression model are analyzed. Figure 7.4 shows the predicted PD against the sequence numbers. Here, it can be seen that the model generally predicts higher values for the first sequence.

Figure 7.5 shows the predicted values against delay changes. The plot visualizes a horizontal pattern at $y = 0$ and a diagonal pattern. These patterns show that the model predicts either a value with approximately the delay change value or zero. This is in line with what is expected, as in a majority of cases, the data includes either PD or SD. Only very few cases are a mix of both. As discussed in Section 4.4.1, it was estimated that in less than 1.85% of cases, both PD and SD occur simultaneously.

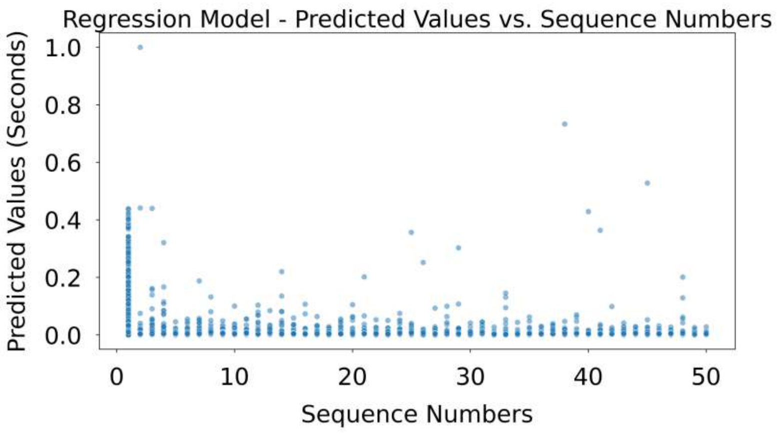


Figure 7.4: Regression Model - Predicted Values vs. Sequence Numbers

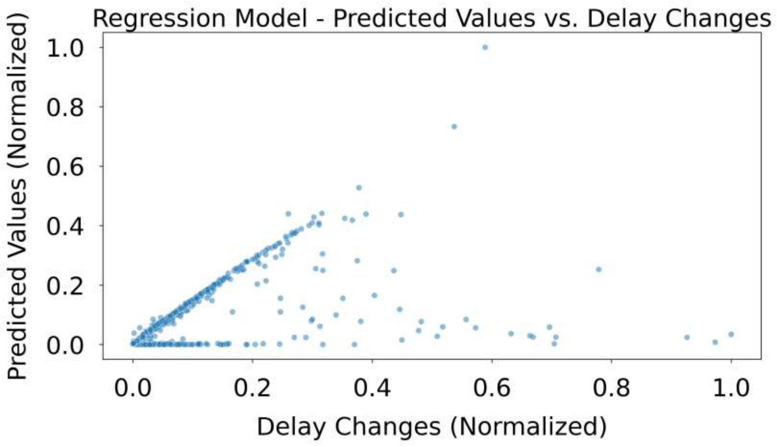


Figure 7.5: Regression Model - Predicted Values vs. Delay Changes

7.1.3 Conclusion of Result Analysis

The analysis of the GNN results for both synthetic and real-world data reveals a significant finding: the primary reason for the poor performance of the GNN on the real-world data is most probably overfitting of the models to the synthetic data. While the proposed architecture demonstrated high accuracy and predictive capability on the synthetic dataset, with a classification accuracy of 96% and a R2 value of 0.8558 for regression, these results did not translate effectively to real-world data. The naive model performed better than the GNN on the real-world dataset, indicating that the GNN may have learned patterns from the synthetic data that do not generalize well to real-world scenarios.

In addition, the distribution of the ratios of PD and SD in the results of the real-world data reflect the 80:20 ratio of the synthetic data and not the estimated real ratio of about 94% PD. This discrepancy strongly suggests that the GNN have adapted too much to the synthetic data and have learned their specific patterns rather than generalizable patterns.

Furthermore, the results show that the proportion of PD for freight trains is higher than for passenger trains, which seems unlikely since passenger trains have higher operational priorities. This finding indicates that the model learns patterns and ratios specific to the synthetic data and, as a result, supports the hypothesis of overfitting.

It is important to note that inaccuracies inherent in the agent-based model inevitably distort the performance metrics of the GNN when applied to real-world data. Given that the agent-based model is not a perfect representation of reality (see Section 7.4.4), the results of the simulation-based validation approach must be interpreted with caution.

7.2 Answering the Research Questions

Drawing from the findings in the previous chapters, the research questions can be answered.

7.2.1 Research Question 1

From what distributions should PD be sampled, and what parameters should be used to calibrate the agent-based train simulation model to reflect real-world delay distributions?

Based on several assumptions and a defined methodology, synthetic data was generated. The exact methodology and results are listed in Chapter 4. A notable finding was that the delay changes in the real-world data follow a Pareto distribution.

Several metrics have been defined to evaluate the similarity between the generated and the real-world data. Based on these metrics and plausibility analyses, the synthetic data shows a similar structure to that of the historical data. However, as discussed in Section 7.1, the GNN overfits the synthetic data and, as a consequence, fails to generalize well to the real-world data. This indicates that the generated data is too dissimilar to the real-world data to be used as a single source for model training. Therefore, the question can be answered as follows: A range of parameters and distributions are suggested as described in Chapter 4. Using the proposed setup for the calibration brings the data close to the real-world scenario. Nevertheless, it's still insufficient as a single source for the training of GNN and, therefore, should be used with caution, awareness of the limitations, and optimally should be further refined.

7.2.2 Research Question 2

How effective are GNN in disaggregating train delays into PD and SD components for synthetic train delay data generated with the agent-based train simulation model?

The results in Section 6.1 and the Discussion 7.1.1 show that the GNN is highly effective in disaggregating train delays into PD and SD components for synthetic train delay data.

7.2.3 Research Question 3

How well does the GNN, trained on the synthetic data, generalize to real-world data for the task of discerning train delays into PD and SD?

The trained GNN have difficulties generalizing to the real-world data and are inferior to the naive model. The exact performance is visible in Section 6.2. In the Discussion 7.1, it is elaborated that the primary reason for the poor performance is most probably overfitting to the synthetic data.

7.3 Contributions to the Railway Domain

To this point, GNN is a rather underutilized method within the railway domain, as shown in the literature review (Section 2.5.1). This is surprising since train networks can efficiently be represented as graphs, which allows the formulation of a range of train-related problems as GNN tasks.

This research extends the current corpus of research of GNN in the railway domain. It is unique in several aspects:

Novel Approach for Disaggregating Train Delays

Within this research, another example is shown of how GNN can be leveraged to solve train delay-related tasks. No other publications have been found so far where GNN were used for the disaggregation of AGD into PD and SD.

Working on Large Scale Railway Networks

What is also unique about this research is the application of GNN on large network graphs, with an average of approximately 280,000 nodes and 1,400,000 edges. In a majority of publications where GNN were applied to train networks (Section 2.5.1), the models were trained on comparably small graphs. This research demonstrates the feasibility of using GNN on large-scale railway networks.

Graph Design and Model Architecture

In the methodology, a graph design was proposed, which was shown to be efficient in translating sequential railway data into a graph structure.

Here, the introduction of supernodes was pivotal since this type of node greatly reduces the number of hops needed to travel from one node to another node.

To this point, no publications have been found that use GatedGCN within the railway domain. This work shows that this specific architecture is functional in capturing the long-range dependencies in the existing graphs.

Summing up, it was shown that the proposed graph design combined with supernodes and GatedGCN is an effective mix of methods to solve train delay propagation problems. This combination enabled the circumvention of difficulties arising from the long-range dependencies in the networks. Hence, those architectural design choices can be recommended for delay management problems.

Conclusion on the Use of Graph Neural Networks in Railway-Related Tasks

From the results on the synthetic data (Section 6.1) it can be derived that GNN are highly suitable for a range of tasks in the railway domain since they seem to be capable of efficiently capturing relevant propagation patterns within the networks. So they could equivalently be used for tasks like, e.g., delay prediction. Since, in this case, a large number of labeled data is available, it can be expected that well-performing models can be trained. Nevertheless, it has to be recognized that the model failed to generalize from the synthetic data to real-world scenarios. Therefore, at this point, it is recommended to apply GNN in the railway domain to tasks where labeled data is available. For cases where no labeled data is available, there should be a strong focus on strategies to improve OOD performance of the GNN.

7.4 Limitations

This section points out the limitations of the applied method and results. It gives an idea of where improvements can be made in future studies and, therefore, builds the bridge to future research.

7.4.1 Data Limitations

Lack of Data

First, it has to be mentioned that this whole study is based on a single day of train delay data. Train delays will show different patterns throughout different days, weeks, months, or whole seasons. Therefore, to build a robust model, a much greater amount of data is of great importance. This would enable thorough data analysis and more sophisticated data sampling, consequently leading to a more representative data set. Also, the lack of data brings the limitation that the model is tested on the real-world data based on which the synthetic data was sampled. Therefore, the model is biased as it indirectly gets information about the test data. The model should optimally be tested on another independent dataset.

It can be concluded that a larger dataset is inevitable to train a reliable and applicable model for the use in real-world scenarios.

Lack of Ground Truth

Another limitation of this study is the lack of actual ground truth. Due to the lack of ground truth, the model has to be trained on synthetic data, which adds multiple sources of error to the training process. Furthermore, the lack of ground truth does not allow straightforward validation, which is why a simulation-based validation was used. Labeled data would allow to train a more precise model and validate it more robustly.

7.4.2 Methodological Assumptions

As described in Section 4.2.2 the data generation is based on several assumptions:

Assumption 1: The distribution of the PD follows the distribution of the delay changes in the historical data.

Assumption 2: The frequencies of PD occurrences and the distributions of PD durations are assumed to be independent of both location and time.

Assumption 3: Frequencies of PD occurrences and the distributions of PD durations are considered independent of OCP type.

These assumptions were established to reduce complexity while still capturing the most important aspects of the data. It is not known if Assumption 1 holds. Consequently, it would be important to investigate this issue more deeply.

Assumptions 2 and 3 were introduced for simplification of the sampling, it can be expected that in real-world scenarios frequencies of PD occurrences and the distributions of PD duration are spatially, temporally, and OCP-type dependent. Due to the data limitations, it was also reasonable to make these assumptions, as more data would be needed to find robust patterns within the data.

It can be expected that more fine-grained data sampling would lead to more realistic data and consequently would enable the training of a more accurate model.

7.4.3 Training of the GNN and Graph Design

As this work is a PoC and the goal was not to train a maximally optimized but a functional model, neither feature selection nor hyperparameter tuning are exhaustive. Both are based on exploratory experiments on small grids combined with different rationales for choosing specific hyperparameters and features. Here, there is room for improvement.

Furthermore, it has to be mentioned that there will be possible improvements in how the graph was designed. There is a wide range of possibilities for designing a graph of railway data. In this work, the approach described in Section 5.4 was used to convert the sequential railway data into a graph. Some design choices were somewhat arbitrary, such as the choice to connect two trains if they visited a station within 180 seconds. A few higher values were tested against this value in an exploratory phase, and the results did not differ significantly. Nevertheless, it has to be pointed out that there might also be room for performance improvements through changes in the graph design choices.

7.4.4 Limitations of the Agent-Based Model

It has to be mentioned that all limitations that are inherent to the agent-based model influence the data generation and consequently also the GNN. A significant limitation of the agent-based model is the inability to sample early departures. To accurately simulate real-world railway scenarios, it is important to give trains the ability to depart earlier than expected, as this happens especially for many freight trains. Another limitation is that the agent-based model processes events instantaneously, meaning that as soon as a section between two stations becomes free, it is immediately occupied by the next waiting train. This approach does not account for the time needed for signal changes, acceleration, or other operational delays, which are crucial in real-world scenarios. Another limitation of the agent-based model was discovered in the sensitivity analysis (Section A.1). It was shown that the agent-based model does not reproduce the expected SD at the start of train journeys. Here, the reasons can be either model or data-related.

7.5 Future Research Directions

Based on the findings and the limitations of this work, several areas of improvement can be suggested for future research.

7.5.1 Larger Dataset

A larger dataset of multiple days of train records is necessary to build a robust and reliable model, as it would give a more representative understanding of how train delay distributions are spread throughout a year. Distributions can differ significantly due to factors like train schedules, seasons, or weather on a given day. A larger dataset is also needed to test the model on data that is independent of the trained model.

It would be optimal to gather labeled data. A few labeled data samples would enable a more robust validation of the model, as this part of the data could be used as a test set. A large labeled dataset would make the agent-based model potentially redundant or at least enable it to generate more representative data.

7.5.2 Improve Calibration of the Agent-Based Model

Despite the thorough calibration (Chapter 4) of the agent-based model, the GNN failed to generalize to the real-world data as described in the discussion of the results (Section 7.1). This shows that the synthetic data is too dissimilar to the real-world data to be used as a single source of training data. Therefore, the calibration of the model has to be improved. It is suggested that the assumptions in Section 4.2.2 are reviewed and tested. Next, it is advised to do a more fine-grained model calibration. Here, there should also be a focus on aiming to have the same ratio of PD to SD in the created data as approximated in Section 4.4.1. It would be optimal to include domain experts from the OEBB and the creators of the agent-based model in the calibration process to get closer to real-world scenarios.

7.5.3 Improvements of the Agent-Based Model

The GNN trained will only be as good as the data it is trained with. As described in the limitations before, constraints within the agent-based model will also be inherited by the GNN.

Therefore, improvements within the agent-based model would also boost the performance of the GNN on the real-world data.

A substantial improvement would be the possibility of enabling early departures. This would allow a more accurate calibration of the model. In the plausibility analysis (Section 4.4), it can be seen that the most considerable discrepancies between the synthetic and the real-world data are in the area of negative delay changes, which might be related to this limitation.

The sensitivity analysis in Section A.1 also unfolded limitations in the model, which should be resolved.

7.5.4 Additional Areas for Improvement

In Table 7.2, additional areas and ideas for improvement of the performance are suggested.

Research Area	Description and Expected Gains
Improving Generalization Capabilities	It is recommended to focus on improving the OOD performance through enhancing the generalization capabilities of the GNN. The literature suggests several methods for this, as discussed in the literature review (see Chapter 2). A data-based approach could be a diversification of the synthetic data, e.g., sampling data with a PD share above and below the estimated share of PD in the estimation, to learn more general delay propagation patterns.
Unsupervised Learning	Another approach to improve performance would be to gather extensive real-world data and pre-train a GNN using the unlabeled data. This could be achieved by a self-supervised pre-training method such as masked feature prediction (e.g. letting the model predict masked delays at random nodes). The pre-trained model could then be fine-tuned with (synthetic) labeled data. This option would allow the GNN to learn delay propagation patterns first from real-world data and subsequently be fine-tuned with synthetic data to learn the actual disaggregation task.
Explainable Artificial Intelligence	Understanding how decisions are made within a neural network is pivotal for improving architectures, training, and performance in the long run. Therefore, XAI is an area that also should get great attention. In comparison to other GNN architectures GatedGCN allow to get a deeper insight into how decisions are made through their gating mechanism. Leveraging this mechanism can uncover which edges or nodes influence predictions most. Through XAI, it can also be estimated how important the various features are. To improve on the model and get a better understanding of how decisions are made, it is highly recommended to also focus on XAI. [14] offers a valuable reference, demonstrating an approach to explaining delay propagation processes using XAI.

Table 7.2: Future Research Directions

CHAPTER 8

Conclusion

8.1 Recap of Research Objectives and Questions

The aim of this thesis was to address the challenge of developing a model that is able to disaggregate AGD into PD and SD. The primary objectives were parameterizing an agent-based simulation model of the Austrian Railway Network, training and evaluating a GNN using synthetically generated data, and finally applying the deep learning model to real-world data to assess its ability to generalize.

8.2 Summary of Key Findings

In the research, despite the lack of ground truth and labeled data, estimates for PD share within the real-world data were approximated. Several distributions were identified and parameterized to sample PD that mimic real-world delay patterns. Using these distributions, the agent-based model was calibrated to create synthetic data that aims to represent real-world railway delay data. The GNN was trained on this generated, where it demonstrated high performance, clearly outperforming the baseline model. The GNN failed to hold these metrics when applied to real-world data, indicating overfitting to the synthetic data and, therefore, the inability to generalize well to OOD.

8.3 Contributions to the Field

1. **Novel Approach to Solve the Disaggregation of Train Delays:** There is little research so far on the problem of discerning AGD into its PD and SD components. The proposed method to use a GNN for this task offers a novel approach.
2. **Innovative Graph Design and Model Architecture:** In the research a graph design is proposed, that offers an effective way to represent sequential railway data

as a graph. Supernodes are utilized combined with GatedGCN to capture long-range dependencies within the railway network. It is shown that the combination of these design choices enables efficient learning of delay propagation within train networks.

3. **Application to Large-Scale Railway Networks:** The study successfully scales the application of GNN to large-scale railway networks. The size of the graphs averages around 280,000 nodes and 1,400,000 edges. It is shown, that GNN can efficiently handle the size of these graphs and make reliable predictions. This is crucial as real-world scenarios are often characterized by large-scale networks.

8.4 Limitations of the Study

1. **Data Constraints:** The research was conducted using only a single day of train delay data. Delays fluctuate based on a variety of factors like seasons, weather, or current timetable, which makes it indispensable to acquire more data for more robust models that generalize well. The data limitation did restrict the possibility of creating realistic data. It also did not allow the testing of the model on an independent real-world dataset.
2. **Simplifying Assumptions:** The study made several assumptions that do not fully encapsulate real-world complexities. One example is the independence of PD occurrence frequencies from location and time. To improve the model, these assumptions should be revisited and refined in subsequent research.
3. **Model Constraints:** All limitations of the agent-based model are also represented in the data and subsequently in the GNN. The agent-based model also builds on simplifying assumptions which lead to patterns different to real-world scenarios. Therefore, further model enhancements are necessary to improve the quality and realism of synthetic data.

8.5 Recommendations for Future Research

Based on the findings, several future research directions are proposed to address the limitations:

1. **Incorporating Larger Datasets:** It would be important to get access to larger datasets, as this would enable a better understanding of delay patterns within the train network. Apart from that, a larger dataset would allow more robust testing.
2. **Refining the Agent-Based Model:** Enhancing the functionality of the agent-based model to more closely resemble real-world scenarios would enable the creation of more realistic data. This, in turn, is expected to improve the performance of the

GNN on real-world data. Furthermore, the calibration method of the agent-based model should be improved, as at this point, the synthetic data proved to be too dissimilar to the real-world data to be used as a single data source for the GNN.

3. **Improve Out-of-Distribution Performance of GNN:** A goal should be to thoroughly work on enhancing the GNN generalization capabilities, as this was the primary weakness of the model in this study. While the model worked well on the synthetic data, it had difficulties generalizing to the real-world data. Improved generalization can be achieved through methods like data augmentation or architectural modifications of the GNN.

8.6 Final Thoughts

Through the creation of a model that can separate AGD of synthetic data into PD and SD, this study marks a substantial development in train delay management. It shows the complexities of applying cutting-edge machine learning techniques in real-world settings, as the models demonstrated strong performance on the generated data but were unable to generalize well to real-world settings. Nevertheless, this work offers valuable information for researchers in the railway domain and establishes ideas for future research, especially in the area of train delay disaggregation. The integration of deep learning models, such as GNN, will be crucial for providing reliable services as they give the opportunity to better understand and predict delay patterns in railway networks.

APPENDIX A

Supplementary Materials

A.1 Sensitivity Analysis

As observed in the data exploration (Section 4.1), freight and passenger trains exhibit different distributions and occurrence frequencies of delay changes at the start of the trip compared to the rest of the journey.

Based on expert opinion, it is hypothesized that a majority of those delays are SD since they typically occur due to delayed resources. Consequently, it is of great interest to test if the agent-based model produces those initial (`train_unique_sequence` = 1) delays, which would indicate the model is a good representation of the real world. To investigate this, an experiment was designed where all parameters except one were set to zero. More precisely, it was tested how raising either the probability of PD occurrence for `train_unique_sequence`! = 1 for freight trains or passenger trains affected the initial delays of the two categories. Experiments were conducted for each parameter using two different seeds, and the results were averaged. The PD were sampled from the Pareto distributions estimated in Section 4.2.3.

In each graph, a horizontal dashed line is shown, which shows the probability in the real-world data that `delay_change` >= 5 minutes occurred at `train_unique_sequence` = 1. The vertical dotted line shows the upper boundary approximated as described in Section 4.2.4. The graph line shows the percentage of trains with a `delay_change` >= 5 minutes at `train_unique_sequence` = 1 for the simulation for different input delay occurrence probabilities.

The graph's boundary and historical values were altered to ensure data anonymity. However, the patterns and relations in the diagrams remained the same. Assuming that delays at `train_unique_sequence` = 1 are only SD, then the sum of the y-values at the intercepts of the blue line and the vertical upper boundary in the two passenger train plots (Figure A.1 and Figure A.2) should ideally match or exceed the historical value indicated by the horizontal dashed line. The same expectation applies to the two freight train plots (Figure A.3 and Figure A.4). The results suggest that while the model's output for passenger trains comes closer to this expected sum, it still slightly undershoots the historical value. The model does not reach the expected SD for freight trains. Figure A.4 shows that passenger train delays have close to no effect on the initial delays of freight trains in the simulation.

Initial Passenger Delays

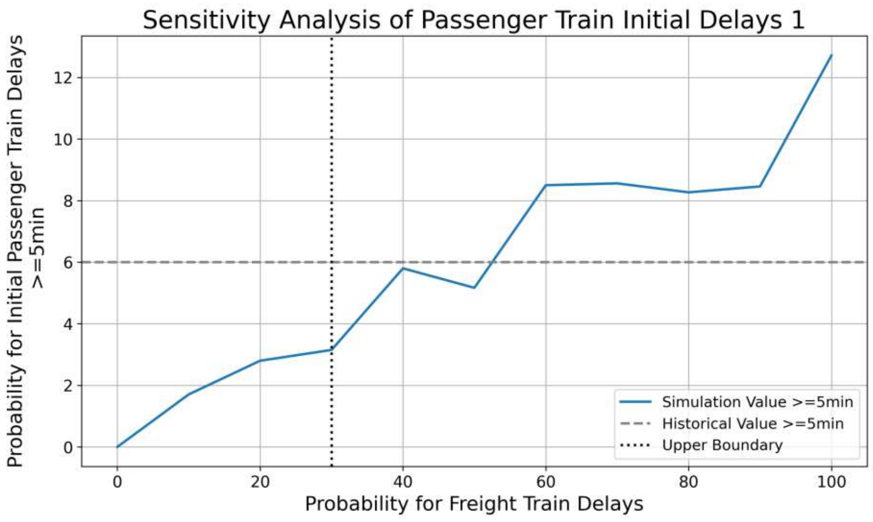


Figure A.1: Sensitivity Analysis of Passenger Train Delays for Increasing Freight Train Parameter

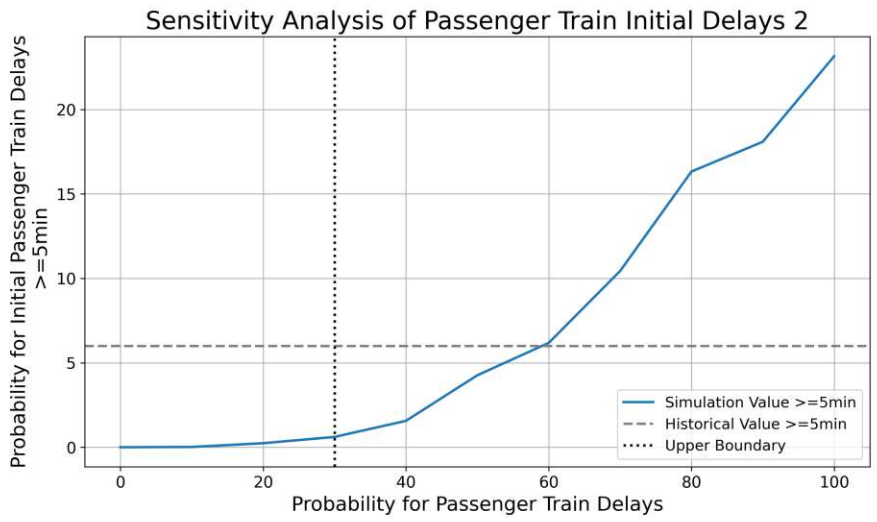


Figure A.2: Sensitivity Analysis of Passenger Train Delays for Increasing Passenger Train Parameter

Initial Freight Delays

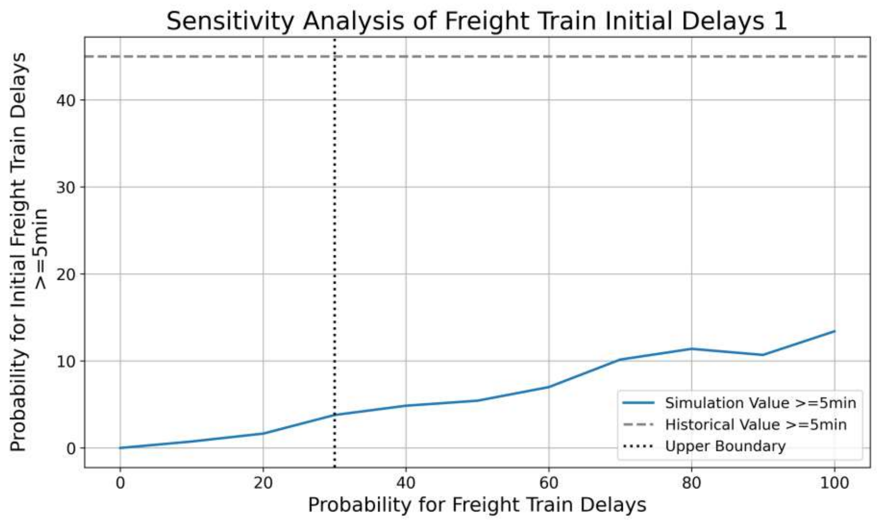


Figure A.3: Sensitivity Analysis of Freight Train Delays for Increasing Freight Train Parameter

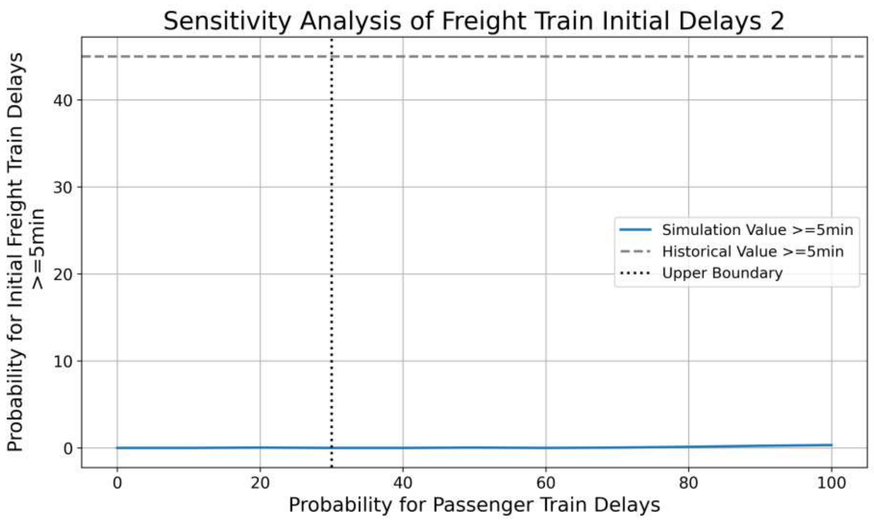


Figure A.4: Sensitivity Analysis of Freight Train Delays for Increasing Passenger Train Parameter

A.2 Feature Evaluation

Feature	**Feature Type**	**Description**
operator_class	categorical	Represents the type of traction unit class.
reference_number	categorical	Represents an Identifier.
uic_number	categorical	International unique identifier for wagons.
order_number	categorical	Represents an Identifier.
train_number	categorical	Unique identifier for each train.
category	categorical	Type of train service.
passenger	bool	Indicates if carrying passengers.
freight	bool	Indicates if carrying freight.
trainpart_weight	int	Weight of the train part in kg.
trainpart_length	int	Length of the train part in meters.
trainpart_speed	int	Train part speed in km/h.
number_of_traction_units	int	Number of traction units.
db640_code	categorical	Identification code for the operating points.
latitude	continuous	Geographic latitude of operating point.
longitude	continuous	Geographic longitude of operating point.
trainpart_id	categorical	Unique identifier for a train part.
sequence_number	int	Position in a sequence for the same trainpart_id. If the train_number stays the same, but the trainpart_id changes at a station, the sequence_number will start counting at 1 again.
ocp_type	categorical	Indicates if a train stopped or passed an OCP.
scheduled_arrival	cyclic	Scheduled arrival time.
scheduled_departure	cyclic	Scheduled departure time.
arrival	cyclic	Actual arrival time.
departure	cyclic	Actual departure time.
arrival_delay_in_seconds	continuous	Arrival delay in seconds.
departure_delay_in_seconds	continuous	Departure delay in seconds.
remarks	string	Additional notes.
export_date	continuous	Data export date.
primary_delay	continuous	Primary delay.
task_id	categorical	Specific task identifier from agent-based model.
sectionID	categorical	Specific section identifier from agent-based model.

Table A.1: Summary of Raw Features with Descriptions

Excluded Feature	**Rationale for Exclusion**
reference_number	Identifier with no predictive relevance.
uic_number	Identifier with no predictive relevance.
order_number	Identifier with no predictive relevance.
freight	Redundant, given binary encoding with ‘Passenger‘
Sequence_Number	Will be replaced by sequence for each train instead of trainpart.
Remarks	
export_date	Metadata not relevant to the predictive model.
task_id	Task-specific identifier within the agent-based model not contributing to prediction.
sectionID	Section identifier within the agent-based model do not contribute to prediction.
scheduled_arrival	Redundant, can be derived from arrival and arrival delay.
scheduled_departure	Redundant can be derived from departure and departure delay.
db640_code	Redundant as the location is already encoded through longitude and latitude.
trainpart_id	Identifier with no predictive relevance.

Table A.2: Summary of Excluded Features and Rationale

Retained Feature	**Rationale for Inclusion**
operator_class	Represents the type of traction unit class, which might affect delays.
passenger	A binary indicator of whether the train is a passenger (true) or freight train (false). Highly relevant for predictions.
category	The category of the train (e.g., regional train, express, long distance) potentially having relevance for predictions due to different prioritization.
latitude	Geographic information is crucial for understanding spatial patterns in train operations.
longitude	Complements latitude.
ocp_type	Has the value stop if a train stops at an operational point; otherwise, it has the value pass. Crucial for predictions.
arrival	Actual arrival time of a train.
departure	Actual departure time of the train.
arrival_delay_in_seconds	High predictive power.
departure_delay_in_seconds	High predictive power.
trainpart_weight	Might have some predictive power regarding to correlation analysis.

Table A.3: Filtered Features for GNN and their Rationale

A.3 Hyperparameter Tuning

A.3.1 Default Parameters

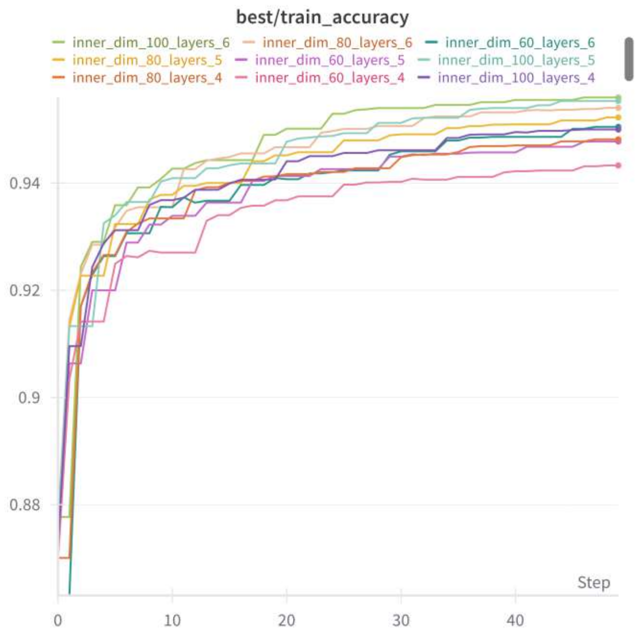
```
metric_best: mae #accuracy
metric_agg: argmin #argmax
dataset:
  task: node
  task_type: regression # classification
  node_encoder: False
  edge_encoder: True
  edge_encoder_name: DummyEdge
train:
  batch_size: 1
model:
  emb_dim_category: 3
  emb_dim_operator_class: 3
  num_operator_classes: 62
  num_categories: 32
  loss_fun: l1  #cross_entropy
gnn:
  head: inductive_node
  layers_pre_mp: 1
  layers_mp: 6
  layers_post_mp: 2
  dim_inner: 40
  layer_type: gatedgcnconv
  act: relu
  residual: True
  dropout: 0.1
  agg: sum
  normalize_adj: False
optim:
  optimizer: adam
  weight_decay: 1e-6
  base_lr: 0.01
  scheduler: reduce_on_plateau
  reduce_factor: 0.5
  schedule_patience: 2
  min_lr: 1e-5
  max_epoch: 20
```

A.3.2 Hyperparameter Tuning Classification

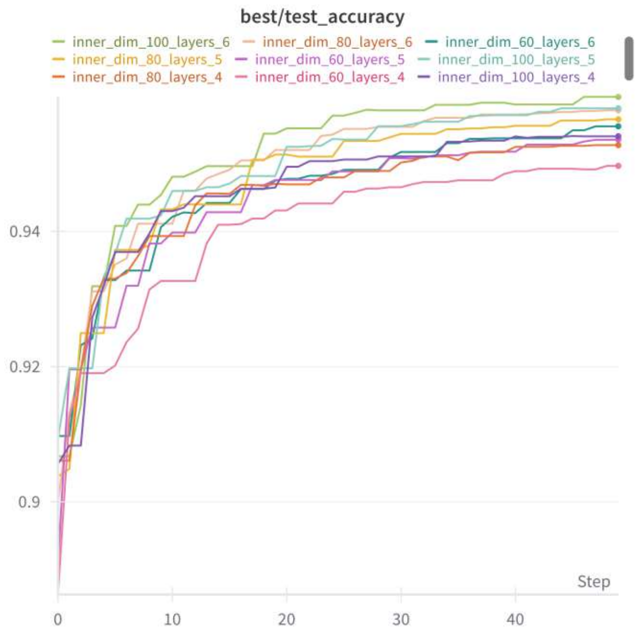
A selection of plots (Figure A.5, Figure A.6 and Figure A.7) of the hyperparameter tuning is shown to give an overview of the influence of hyperparameters on the target variables. The plots of the validation dataset are omitted since they are highly similar to the test set. Also, a Table A.4 of the best ten hyperparameter results is shown.

layers	**dim_inner**	**params**	**accuracy**	**precision**	**recall**	**f1**	**auc**
6	100	316601.0	**0.9599**	**0.9736**	0.9761	**0.9748**	**0.9888**
5	100	265701.0	0.9582	0.9731	0.9744	0.9737	0.9880
6	80	203681.0	0.9579	0.9708	**0.9764**	0.9736	0.9878
5	80	170961.0	0.9566	0.9706	0.9749	0.9728	0.9870
6	60	115561.0	0.9555	0.9689	0.9754	0.9721	0.9865
4	100	214801.0	0.9541	0.9687	0.9737	0.9712	0.9858
5	60	97021.0	0.9536	0.9667	0.9752	0.9709	0.9853
3	80	105521.0	0.9448	0.9594	0.9717	0.9655	0.9795
3	100	163901.0	0.9456	0.9624	0.9694	0.9659	0.9805
3	60	59941.0	0.9415	0.9560	0.9710	0.9635	0.9769

Table A.4: Best Hyperparameter Tuning Results for Classification

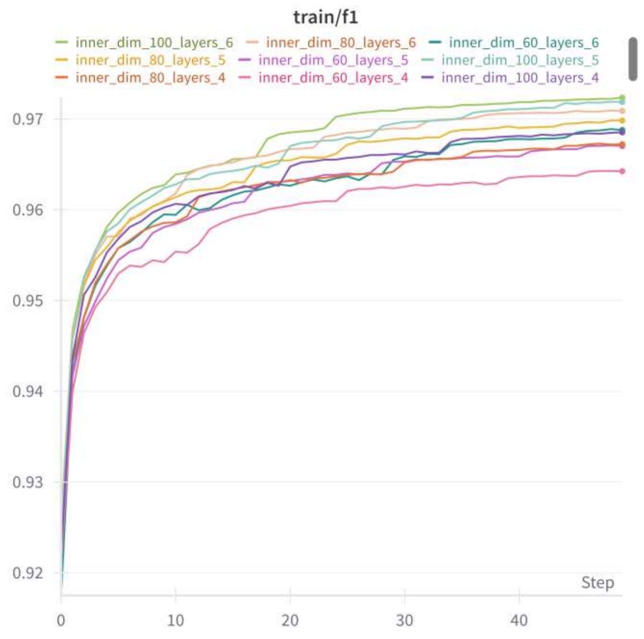


(a) Training Accuracy

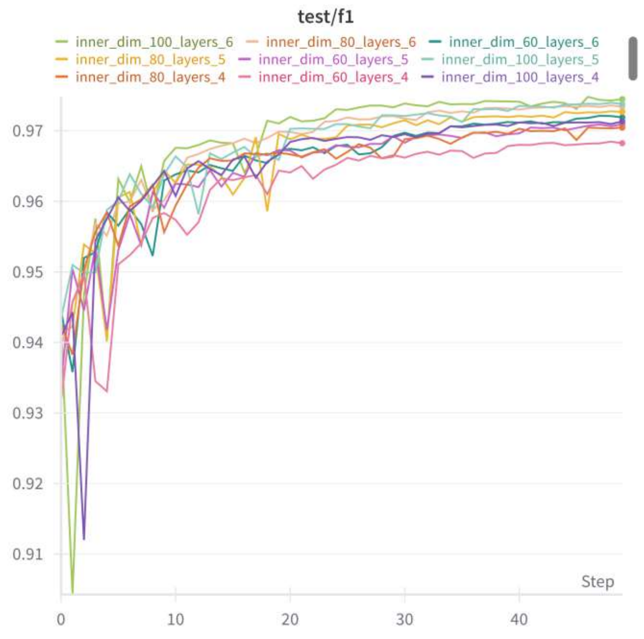


(b) Test Accuracy

Figure A.5: Hyperparameter Tuning Results for the Classification Task (Part 1/3)

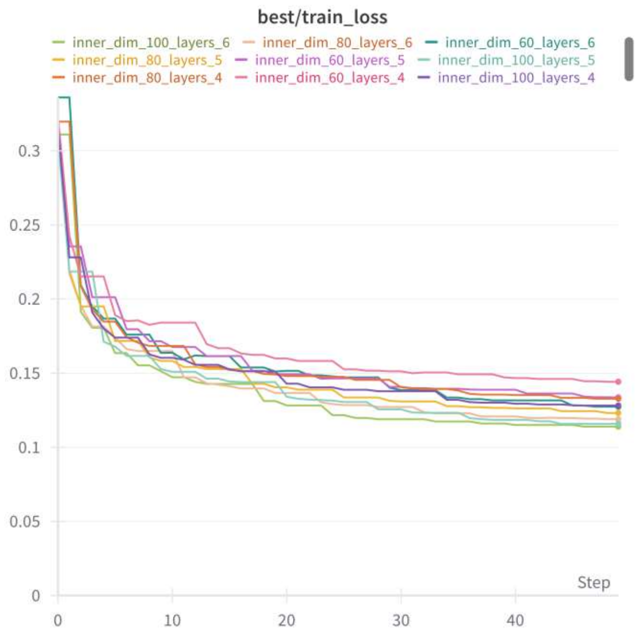


(a) Training F1 Score

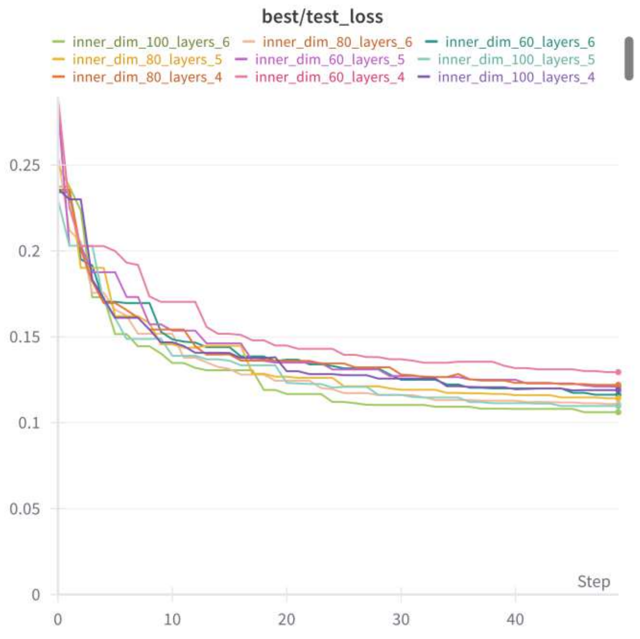


(b) Test F1 Score

Figure A.6: Hyperparameter Tuning Results for the Classification Task (Part 2/3)



(a) Training Loss



(b) Test Loss

Figure A.7: Hyperparameter Tuning Results for the Classification Task (Part 3/3)

A.3.3 Hyperparameter Tuning Regression

layers	dim_inner	mae	mse	r2	spearmanr	params
5	40	**0.04316**	0.15086	0.85278	0.87187	43881.0
5	55	0.04453	0.14804	0.85554	0.8736	81786.0
5	70	0.04467	0.14416	0.85932	**0.88619**	131391.0
9	55	0.04466	0.15992	0.84395	0.8681	144266.0
9	70	0.04488	0.14794	0.85564	0.88238	231911.0
7	55	0.04556	0.15273	0.85096	0.87234	113026.0
7	70	0.04752	**0.14156**	**0.86186**	0.87915	181651.0
7	40	0.04891	0.15379	0.84993	0.86925	60601.0
9	40	0.05013	0.19556	0.80917	0.87499	77321.0

Table A.5: Best Hyperparameter Tuning Results for Regression

A.3.4 Classification Task - Final Model

Metric	Train Set	Validation Set	Test Set
Epochs	49	49	49
Accuracy	0.95767	0.96005	0.96016
Precision	0.97112	0.97269	0.9732
Recall	0.97529	0.97728	0.97678
F1 Score	0.9732	0.97498	0.97499
AUC	0.98848	0.98916	0.989
Number of Parameters	316601	316601	316601

Table A.6: Results for Final Classification Model

A.3.5 Regression Task - Final Model

Metric	Test Set	Validation Set	Train Set
Epochs	49	49	49
Number of Parameters	43881	43881	43881
Mean Absolute Error (MAE)	0.045	0.04629	0.04719
Mean Squared Error (MSE)	0.1478	0.15162	0.16853
R-Squared (R2)	0.85577	0.84509	0.83147
Spearman's Rank Correlation	0.87477	0.8745	0.848
Root Mean Squared Error (RMSE)	0.38445	0.38939	0.41052

Table A.7: Results for Final Regression Model

A.4 Overview of Tools Used

I declare that I used the generative AI tool ChatGPT (Version GPT-4o) in this thesis, but only to improve the writing style and grammar. All the research, analysis, and content were done by me. The AI was only used to refine the language of text I had already written without changing the content or relying on information from the AI.

List of Figures

List of Tables

List of Acronyms

AGD	Aggregated Delays. 1–5, 8, 9, 11, 13, 24, 31, 32, 34, 54, 57, 59, 65, 71, 73
ANN	Artificial Neural Network. 5, 6, 11–13, 43
CNN	Convolutional Neural Networks. 18
GatedGCN	Gated Graph Convolutional Networks. 6, 12, 15, 18, 66, 70, 72
GCN	Graph Convolutional Networks. 12, 17, 18
GNN	Graph Neural Network. 2–13, 15–18, 39–41, 43, 46, 47, 49–55, 57, 59, 60, 63–66, 68–73, 89
K-S Test	Kolmogorov-Smirnov Test. 26, 30–34, 37, 91
KL-divergence	Kullback–Leibler divergence. 30, 31
MAE	Mean Absolute Error. 53, 60
MSE	Mean Squared Error. 53
NILM	Nonintrusive Load Monitoring. 11
OCP	Operational Control Point. 25, 29, 39, 42–44, 67, 79
OEBB	Austrian Federal Railways. 2, 3, 5, 19, 29, 69
OOD	Out-of-Distribution. 13, 66, 70, 71
PD	Primary Delays. 1–9, 11, 13, 24, 25, 29, 30, 36, 37, 40, 43, 45, 49, 50, 52–55, 59–65, 67, 69–73, 75
PoC	Proof of Concept. 41, 50, 67
R2	R-squared. 53, 56, 60, 63

SD Secondary Delays. 1–5, 7–9, 11, 13, 24, 36, 37, 40, 49, 50, 52–54, 57, 59–65, 68, 69, 71, 73, 75, 76

WAPE Weighted Absolute Percentage Error. 55, 56

XAI Explainable Artificial Intelligence. 8, 70

Bibliography

[1] (2023) Fast jeder dritte reisende kam 2023 verspätet an. Tagesschau. Accessed: 2024-08-11. [Online]. Available: https://www.tagesschau.de/wirtschaft/unternehmen/deutsche-bahn-verspaetung-100.html

[2] D. Rößler, J. Reisch, F. Hauck, and N. Kliewer, "Discerning primary and secondary delays in railway networks using explainable ai," *Transportation Research Procedia*, vol. 52, pp. 171–178, 2021, 23rd EURO Working Group on Transportation Meeting, EWGT 2020, 16-18 September 2020, Paphos, Cyprus. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352146521000405

[3] M. Rößler, M. Wastian, A. Jellen, S. Frisch, D. Weinberger, P. Hungerländer, M. Bicher, and N. Popper, "Simulation and optimization of traction unit circulations," in *2020 Winter Simulation Conference (WSC)*, 2020, pp. 90–101.

[4] A. Yamamura, M. Koresawa, S. Adachi, and N. Tomii, "Identification of causes of delays in urban railways," *Computers in Railways*, vol. 13, pp. 403–414, 2013.

[5] J. Manitz, J. Harbering, M. Schmidt, T. Kneib, and A. Schöbel, "Source estimation for propagation processes on complex networks with an application to delays in public transportation systems," *Journal of the Royal Statistical Society Series C: Applied Statistics*, vol. 66, no. 3, pp. 521–536, 2017.

[6] Y. Ochiai, Y. Shibata, and N. Tomii, "An algorithm to identify delay propagation routes based on visualization of asso-ciation rules," in *Proceedings of the 2nd International Railway Symposium Aachen*, 2019, p. 245.

[7] C.-W. Palmqvist, I. Johansson, and H. Sipilä, "A method to separate primary and secondary train delays in past and future timetables using macroscopic simulation," *Transportation Research Interdisciplinary Perspectives*, vol. 17, p. 100747, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S259019822200207X

[8] C. Wen, Z. Li, J. Lessan, L. Fu, P. Huang, and C. Jiang, "Statistical investigation on train primary delay based on real records: evidence from wuhan–guangzhou hsr," *International Journal of Rail Transportation*, vol. 5, pp. 1–20, 03 2017.

[9] C. Wen, Z. Li, P. Huang, J. Lessan, L. Fu, and C. Jiang, “Cause-specific investigation of primary delays of wuhan–guangzhou hsr,” *Transportation Letters*, vol. 12, no. 7, pp. 451–464, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1942786722001989

[10] W. Schwanhäußer, “Die bemessung der pufferzeiten im fahrplangefüge der eisenbahn,” PhD thesis, RWTH Aachen University, Aachen, Germany, 1974.

[11] Z. Li, P. Huang, C. Wen, and F. Rodrigues, “Railway network delay evolution: A heterogeneous graph neural network approach,” 2023.

[12] J. S. Heglund, P. Taleongpong, S. Hu, and H. T. Tran, “Railway delay prediction with spatial-temporal graph convolutional networks,” in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–6.

[13] D. Zhang, Y. Peng, Y. Zhang, D. Wu, H. Wang, and H. Zhang, “Train time delay prediction for high-speed train dispatching based on spatio-temporal graph convolutional network,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 3, pp. 2434–2444, 2022.

[14] P. Huang, J. Guo, S. Liu, and F. Corman, “Explainable train delay propagation: A graph attention network approach,” *Transportation Research Part E: Logistics and Transportation Review*, vol. 184, no. C, 2024.

[15] D. Zhang, Y. Xu, Y. Peng, C. Du, N. Wang, M. Tang, L. Lu, and J. Liu, “An interpretable station delay prediction model based on graph community neural network and time-series fuzzy decision tree,” *IEEE Transactions on Fuzzy Systems*, vol. PP, pp. 1–13, 01 2022.

[16] W. Jiang and J. Luo, “Graph neural network for traffic forecasting: A survey,” *CoRR*, vol. abs/2101.11174, 2021. [Online]. Available: https://arxiv.org/abs/2101.11174

[17] P. A. Schirmer and I. Mporas, “Non-intrusive load monitoring: A review,” *IEEE Transactions on Smart Grid*, vol. 14, no. 1, pp. 769–784, 2023.

[18] R. Shang, S. Chen, Z. Chen, and C.-T. Lu, “Graphnilm: A graph neural network for energy disaggregation,” in *Advances in Knowledge Discovery and Data Mining*, D.-N. Yang, X. Xie, V. S. Tseng, J. Pei, J.-W. Huang, and J. C.-W. Lin, Eds. Singapore: Springer Nature Singapore, 2024, pp. 431–443.

[19] ÖBB, “Zahlen daten fakten 2019/20,” 2020, accessed: 2024-08-14. [Online]. Available: https://konzern.oebb.at/dam/jcr:b17c14a2-d8a3-4d3c-8a40-912cbeefa6ab/OEBB_Zahlen_2020-2_de_web.pdf

[20] U. Alon and E. Yahav, “On the bottleneck of graph neural networks and its practical implications,” 2021.

[21] V. P. Dwivedi, L. Rampášek, M. Galkin, A. Parviz, G. Wolf, A. T. Luu, and D. Beaini, “Long range graph benchmark,” 2023.

[22] J. Tönshoff, M. Ritzert, E. Rosenbluth, and M. Grohe, “Where did the gap go? reassessing the long-range graph benchmark,” 2023.

[23] Z. Zhong, C.-T. Li, and J. Pang, “Hierarchical message-passing graph neural networks,” *Data Mining and Knowledge Discovery*, vol. 37, no. 1, pp. 381–408, 2023.

[24] R. Liu, P. Calafiura, S. Farrell, X. Ju, D. T. Murnane, and T. M. Pham, “Hierarchical graph neural networks for particle track reconstruction,” 2023.

[25] S. I. Nikolenko, *Synthetic data for deep learning.* Springer, 2021, vol. 174.

[26] L. Xu, H. Liu, B. Xiao, X. Luo, DharmarajVeeramani, and Z. Zhu, “A systematic review and evaluation of synthetic simulated data generation strategies for deep learning applications in construction,” *Advanced Engineering Informatics*, vol. 62, p. 102699, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1474034624003471

[27] G. Lombardo, M. Pellegrino, A. Poggi *et al.*, “Unsupervised continual learning from synthetic data generated with agent-based modeling and simulation: a preliminary experimentation.” in *WOA*, 2022, pp. 116–126.

[28] C. M. de Melo, A. Torralba, L. Guibas, J. DiCarlo, R. Chellappa, and J. Hodgins, “Next-generation deep learning based on simulators and synthetic data,” *Trends in cognitive sciences*, vol. 26, no. 2, pp. 174–187, 2022.

[29] H. Li, X. Wang, Z. Zhang, and W. Zhu, “Out-of-distribution generalization on graphs: A survey,” *ArXiv*, vol. abs/2202.07987, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:246867220

[30] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.

[31] B. Sanchez-Lengeling, E. Reif, A. Pearce, and A. B. Wiltschko, “A gentle introduction to graph neural networks,” *Distill*, 2021, https://distill.pub/2021/gnn-intro.

[32] X. Bresson and T. Laurent, “Residual gated graph convnets,” 2018. [Online]. Available: https://arxiv.org/abs/1711.07553

[33] A. Gupta, P. Matta, and B. Pant, “Graph neural network: Current state of art, challenges and applications,” *Materials Today: Proceedings*, vol. 46, pp. 10 927–10 932, 2021, international Conference on Technological Advancements in Materials Science and Manufacturing. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214785321010543

[34] K. O'Shea and R. Nash, "An introduction to convolutional neural networks," 2015. [Online]. Available: https://arxiv.org/abs/1511.08458

[35] A. Kolmogorov, "Sulla determinazione empirica di una legge di distribuzione," *Giornale dell'Istituto Italiano degli Attuari*, vol. 4, pp. 83–91, 1933.

[36] N. V. Smirnov, "Table for estimating the goodness of fit of empirical distributions," *The Annals of Mathematical Statistics*, vol. 19, no. 2, pp. 279–281, 1948.

[37] OEBB-Infrastruktur AG, "Schienennetz-nutzungsbedingungen 2024," 2024, page 34, Accessed: 2024-07-29. [Online]. Available: https://infrastruktur.oebb.at/de/geschaeftspartner/schienennetz/snnb/snnb-2024/schienennetz-nutzungsbedingungen-2024.pdf

[38] S. Kullback and R. A. Leibler, "On information and sufficiency," *The Annals of Mathematical Statistics*, vol. 22, no. 1, pp. 79–86, 1951.

[39] J. You, R. Ying, and J. Leskovec, "Design space for graph neural networks," 2021. [Online]. Available: https://arxiv.org/abs/2011.08843

[40] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," 2019. [Online]. Available: https://arxiv.org/abs/1903.02428