Technical University of Munich

Department of Informatics

Development of a Deep Learning Surrogate for the Four-Step Transportation Model

Master's Thesis in Data Engineering and Analytics

Nikita Makarov

Development of a Deep Learning Surrogate for the Four-Step Transportation Model

Entwicklung eines Deep Learning Surrogats für das vierstufige Transportmodell

A thesis presented as part of the requirements of the Degree of Master of Science in Data Engineering and Analytics at the Department of Informatics, Technical University of Munich.

Advisor	M. Sc. Santhanakrishnan Narayanan
Supervisor	Univ.-Prof. Dr. Constantinos Antoniou Chair of Transportation Systems Engineering
Submitted by	Nikita Makarov
Submitted on	Munich, 15.12.2021

Declaration

I confirm that this master's thesis is my own work and I have documented all sources and materials used.

This thesis has not previously been submitted elsewhere for purposes of assessment.

Makarov

Munich, 15.12.2021, Signature

Abstract

Transport planning is a critical component of the transport system, especially to understand how large projects will affect a city or region over multiple decades. The currently most used method, the Four Step Model, has been developed over many decades and is able to solve many important problems. However, it has a number of downsides, mainly that it unrealistically assumes that travellers perform decisions sequentially and that the method cannot entirely automatically tune the parameters to fit observed data. Graph neural networks are identified as a suitable candidate to overcome the shortcomings but require an impractical large amount of observed data. Thus, a surrogate model procedure is proposed and implemented, whereby data is generated by extracting information either from the real world or synthetic networks, and an existing Four Step Model is applied to retrieve the usage for private and public transportation. The transport usage results are then used as targets to train graph neural networks. From the machine learning point of view, the task is defined as transfer learning with either node classification or regression for the transport usage level. Such a task is not yet found in existing literature, and the novelty of the thesis lies in the development of the proof of concept of the surrogate model. Three graph neural networks are examined, GCNII, a novel extension to the GATv2 model, as well as a combined GCNII and GAT model. The three models are applied to various experiments to understand their limits and potential as transport planning surrogates. It is observed that the graph neural networks can successfully model the basic surrogate setup whilst overcoming the downsides of the Four Step Model, but require more data to be accurate enough for practical usage. Additionally, all graph neural networks tested suffer worse performance with an increase in network size. Future research should dive deeper into creating larger datasets, as well as to propose new graph neural network approaches which are able to generalize to arbitrary large networks.

Acknowledgements

Throughout this thesis, I received an incredible amount of support, for which I am grateful. First and foremost, I would like to express my gratitude to my supervisors, Santhanakrishnan Narayanan and Prof. Dr. Constantinos Antoniou, for accepting and supporting my topic throughout the entire thesis. I would especially like to thank Santhanakrishnan Narayanan for being open to unpredictable ideas, having patience explaining key transport topics to me and inspiring me to push for more. Additionally, thank you to Dr. Carlos Llorca Garcia for supporting the idea and providing critical resources. Finally, I would like to thank my friends and family for being with me during the most challenging times, especially with an ongoing pandemic. I would like to dedicate this thesis to my parents and my wife Christine for their endless support and love.

Contents

1 Introduction **13**
- 1.1 Motivation . . . 13
- 1.2 Research Questions, Objectives & Framework . . . 14
- 1.3 Contributions & Thesis Structure . . . 15

2 Literature Review **17**
- 2.1 Strategic Transport Planning & Four Step Model . . . 17
  - 2.1.1 Four Step Model . . . 17
  - 2.1.2 Downsides of the Four Step Model . . . 18
- 2.2 Existing Work Solving the Four Step Model's Problems . . . 19
  - 2.2.1 Combining Individual Steps Together . . . 19
  - 2.2.2 Machine Learning in the Four Step Model . . . 21
  - 2.2.3 Uncertainty Quantification in the Four Step Model . . . 22
  - 2.2.4 Automatic Calibration . . . 23
- 2.3 End to End Deep Learning for Transport Planning . . . 23
- 2.4 Surrogate Modelling . . . 24
- 2.5 Summary & Thesis Focus . . . 26
  - 2.5.1 Summary . . . 26
  - 2.5.2 Thesis Focus . . . 27

3 Surrogate Models **28**
- 3.1 General Training Procedure, Classification & Regression . . . 28
- 3.2 Baseline Models . . . 28
  - 3.2.1 Majority Classifier . . . 29
  - 3.2.2 Mean Regressor . . . 29
  - 3.2.3 Random Forests . . . 29
- 3.3 Notation . . . 29
- 3.4 Deep Learning . . . 33
  - 3.4.1 Basic Functionality . . . 33
  - 3.4.2 Deep Learning Training . . . 34
  - 3.4.3 Deep Learning Baseline . . . 35
- 3.5 Graph Neural Networks . . . 35
  - 3.5.1 Graph Convolutional Neural Networks . . . 35
  - 3.5.2 GCNII . . . 37
  - 3.5.3 Graph Attention Networks . . . 37
  - 3.5.4 GATv3 . . . 39
  - 3.5.5 Combined GAT and GCN . . . 40

3.6 Regularization 42
3.6.1 Dropout 43
3.6.2 GraphNorm 43
3.7 Loss Functions and Evaluation Metrics 43
3.7.1 Loss Functions 43
3.7.2 Classification Evaluation Metrics 44
3.7.3 Regression Evaluation Metrics 45
4 Problem Formalization & Surrogate Model Framework **48**
4.1 Surrogate for the Four Step Model 48
4.2 Formal definition 49
4.3 Surrogate Approach & Scope of the Thesis 50
4.3.1 Surrogate Model Definition 50
4.3.2 Abstract Surrogate Model Algorithm 51
4.3.3 Assumptions & Methodological Scope 52
5 Data Generation & Transformation **54**
5.1 Data Generation 54
5.1.1 Augmented Region Generation 54
5.1.2 Synthetic Region Generation 56
5.1.3 Four Step Model Implementation 58
5.2 Data Transformation 59
5.2.1 Data Transformation Pipeline 60
5.2.2 Graph Transformations 61
5.3 Implementation 62
6 Experimental Setup **64**
6.1 Basic Transport Planning Surrogate 65
6.1.1 Line Graph vs Direct Graph Conversion 65
6.1.2 Joint Public and Private Transportation Modelling 67
6.1.3 Comparison of GCNII, GATv2, GATv3 and GCN+GATv3 67
6.2 Regression Predictions 68
6.2.1 GNN Regression 68
6.2.2 Hierarchical Regression 70
6.3 Fine Grained Classification 72
6.3.1 Optimal Bucket Strategy 73
6.3.2 Validating Non-Linear Buckets & Additional Synthetic Data 74
7 Experimental Results **76**
7.1 Basic Transport Planning Surrogate Model Results 77
7.1.1 Line Graph vs Direct Graph Conversion Results 77
7.1.2 Joint Public and Private Transportation Modelling Results 77
7.1.3 Comparison of GCNII, GATv2, GATv3 and GCN+GATv3 Results . . . 78
7.1.4 Output Examples 80
7.1.5 Basic Surrogate Findings Overview 82
7.2 Regression & Hierarchical Regression Results 83

7.3 Fine Grained Classification Results 85
7.3.1 Optimal Bucket Strategy Results 85
7.3.2 Validating Non-Linear Buckets & Additional Synthetic Data - Results . 88
7.3.3 Overall Fine Grained Classification Findings 90
8 Discussion **91**
8.1 Answer to Primary Research Question 91
8.1.1 Functioning Aspects 91
8.1.2 Non-functioning Aspects 92
8.1.3 Primary Research Question Conclusion 92
8.2 Insights for Graph Neural Networks 93
8.3 Use Cases for Existing Surrogate Models 94
9 Conclusion **95**
9.1 Overview 95
9.2 Future Research 96
9.2.1 Graph Neural Network Surrogate for the Four Step Model 97
9.2.2 Application of Graph Neural Networks in Transport Planning 97
9.2.3 Graph Neural Network Research 98
A Data Transformation Module Categories **99**

List of Figures

1 Research framework of the thesis. 15
2 A simplified comparison of the proposed method in this thesis to existing literature. 27
3 A visualisation of how a neuron works. 34
4 The inner workings of a single GCN layer. 36
5 The inner workings of a single GAT layer. 39
6 An example of a true network graph (left) vs zone graph (right), rendered in PTV Visum 2021. 41
7 A visualisation of the setup for the combined GCN and GAT model. 41
8 How a deep learning surrogate model would be used in practise. 49
9 Augmented dataset histogram of graph sizes. 55
10 Example of a single sample from the augmented dataset. 57
11 Synthetic dataset histogram of graph sizes. 58
12 UML class diagram of how the data transformation steps are implemented. . . . 61
13 Illustration of the hierarchical regression model. 72
14 Training loss of cross entropy of the GCNII model over training epochs. 79
15 F1 score of the GCNII model over the validation dataset. 79
16 Prediction of GCN+GAT model. 80
17 Targets using Visum simulation. 80
18 Prediction II of GCN+GAT model. 80
19 Targets II using Visum simulation. 80
20 Prediction of GCN+GAT model, able to predict alternate routing. 81
21 Targets using Visum simulation on congested network. 81
22 Example of probabilities of prediction for a link. 82
23 Predicted output vs true output for the GCNII - Buckets-e90 - extra data model on the test PrT dataset. 87
24 Predicted output vs true output for the GCNII - Buckets-nl54 - extra data model on the test PrT dataset. 87
25 Validation $MAE_{\geq 10}$ in relation to the number of buckets on PrT classification for the GCNII model. 88
26 $MAE_{\geq 10}$ with respect to graph size. 90
27 Validation $MAE_{\geq 10}$ in relation to training time. 93

List of Tables

2 Comparison of the GNNs used in this thesis. 42
3 Comparison of the used metrics and losses. 47
4 Comparison of the Line Graph and Direct Graph transformations. 62
5 Overview of all experimental setups. 64
6 Overview of all experimental results. 76
7 Results on the basic PrT classification task comparing line graph with direct graphs on the validation dataset. 77
8 Results on test datasets for basic separate PrT and PuT modelling, using more advanced models. 78
9 Results of regression on validation datasets for PrT. 84
10 Results of fine grained classification on validation datasets for PrT. 86
11 Results of additional synthetic training data for fine grained classification. . . . 89

Glossary

Numbers

4SM - Four Step Transport Model.......... 13

C

CE - Cross Entropy Error.......... 44

G

GAT - Graph Attention Neural Network.......... 37

GATv3 - Graph Attention Neural Network with residual connections to input.......... 40

GCN - Graph Convolutional Neural Network.......... 35

GCN + GAT - Combined GCN and GAT model, focus on zone information.......... 41

GCNII - Graph convolution neural network with modifications to allow for deep models. 37

GNN - Graph neural network.......... 35

M

MAE - Mean Absolute Error.......... 45

MLP - Multilayer perceptron.......... 35

MSE - Mean Squared Error.......... 44

P

PrT - Private transportation, in this context, car.......... 59

PuT - Public transportation, in this context, bus.......... 59

T

tanh - Hyperbolic tangent function.......... 33

TAZ - Traffic Analysis Zone.......... 17

V

veh/h - Vehicles per hour.......... 55

1. Introduction

The introduction chapter starts off with the motivation, before continuing on to the research questions, objectives and framework. Finally, the contributions and thesis structure are presented.

1.1. Motivation

Transportation plays a critical role in modern society. For example, the commute time is shown to be the strongest factor affecting whether a family can escape poverty (Chetty, Hendren, et al. 2015), showing that it is essential that transportation projects be planned out well. Focusing on long term travel modelling, various tools have been developed over the decades, with the Four Step Model (4SM) being the most popular approach. The method models both the supply and demand aspects, however, it has a number of known drawbacks, which will be examined in detail in later chapters, but a short summary is provided here. A major drawback has been that the model has strong assumptions on the underlying decision-making process, such as stating that the destination is chosen irrespectively of the transport mode. Various improvements have been proposed to improve the modelling capacity, but the underlying problems persist. Another major drawback is that the Four Step Model outputs single values, not statistical distributions. Due to all models having a certain degree of error to them, it is harder for decision-makers to trust the method's output (Rasouli and Timmermans 2012). Finally, the model requires a lengthy and manual calibration process to make them similar to true, observed data. Automatic calibration approaches have been proposed, but few examine the end to end calibration, and those that do require a high amount of computational power, thus making them limited in their practical application.

Looking at transport systems, new modes are emerging, changing the underlying patterns of how users make choices (Antoniou, Efthymiou, and Chaniotakis 2019). Additionally, the popularity of mobile phones and information platforms has widespread effects on travel choices (Zito et al. 2011). Since early 2020, COVID-19 has also drastically affected the number of trips people are taking and the modes that they feel safe in (Vos 2020). The combination of these effects shows that quickly adaptable transport planning systems are needed for the coming decades. On the other hand, both more data and computational power is becoming available to transport planning, meaning that it is now possible to employ more data driven methods (Zannat and Choudhury 2019). The main underlying difference of data driven methods, in comparison to model based methods, is that the exact model formulation is learnt from the data and the method can thus exploit patterns which would be overlooked if done otherwise. In data driven methods, machine learning has become the prevailing class of methods (Efron and Hastie 2016). Thus, there has been a growing popularity of machine learning in transport planning (O. Iliashenko, V. Iliashenko, and Lukyanchenko 2021).

In the field of machine learning, the deep learning subclass of methods have been breaking previous baselines in several fields, ranging from image recognition to natural language processing (Pouyanfar et al. 2019). The main advantage of deep learning methods is their ability to model highly non-linear processes and being universal approximators to any given function. Since 2015, a new class of deep learning methods have appeared which apply deep learning functions on graphs, called graph neural networks. As transport networks can be easily interpreted as graphs, it is a natural application for graph neural networks. However, until now, their application in transportation has been mostly limited to short term traffic forecasting (Manibardo, Laña, and Ser 2021).

Examining simulation sciences, a sub-field called *Surrogate Modelling* has been rising where instead of running simulation functions directly, a second function is built to mimic the original function, the so-called surrogate model (Sudret, Marelli, and Wiart 2017). The surrogate model possesses some properties that the original function does not have, such as being computationally cheap. As machine learning models are often faster to evaluate and can approximate various functions, they have often been employed as surrogates (L. Sun et al. 2020). Due to GPUs and matrix-based operations, deep learning surrogates are able to perform highly efficiently. In a classification based scenario, deep learning models can also output probability scores for each category while training directly from real, observed data. Specifically, graph neural network surrogates have recently reached the state of the art levels in certain physics fields, such as cloth simulations (Pfaff et al. 2020).

Comparing the issues of the existing transport planning methods with the potential of a graph neural network-based surrogate, it can be seen that there is a significant overlap. Therefore, this thesis attempts to explore whether such a surrogate function can be developed for the 4 Step Model and where the limits currently exist.

1.2. Research Questions, Objectives & Framework

The primary research question of the thesis is defined as following:

Can a proof of concept deep learning surrogate be built for the 4 Step Transportation model for networks with a maximum of 80 nodes?

The research framework taken in this thesis is visualized in Figure 1. The main steps are to first perform a literature review to identify the main downsides of the Four Step Model as well as to understand the current state of the art surrogate models. Next the research objective is formulated with the requirements to be fulfilled, and a potential solution formally defined. The following step generates the first dataset, with which the initial experiments are conducted. Three experiments sections are found sequentially. To summarize, the first experiment block tries to understand whether a basic surrogate model can be built, the second section looks at how to make the surrogate precise in a regression setting and the final experiments attempt to see whether classification can improve predictions. Each experiment section has multiple experiments, with each experiment performed by first applying the data processing, implementing the respective model, then training the model before finally evaluating on unseen data and performing further analysis. After all experiments are finished, the final analysis, discussion and conclusion are performed.

The aim of the thesis is to understand what the existing limits are for surrogate modelling for the 4 Step Model, and which direction should be followed.

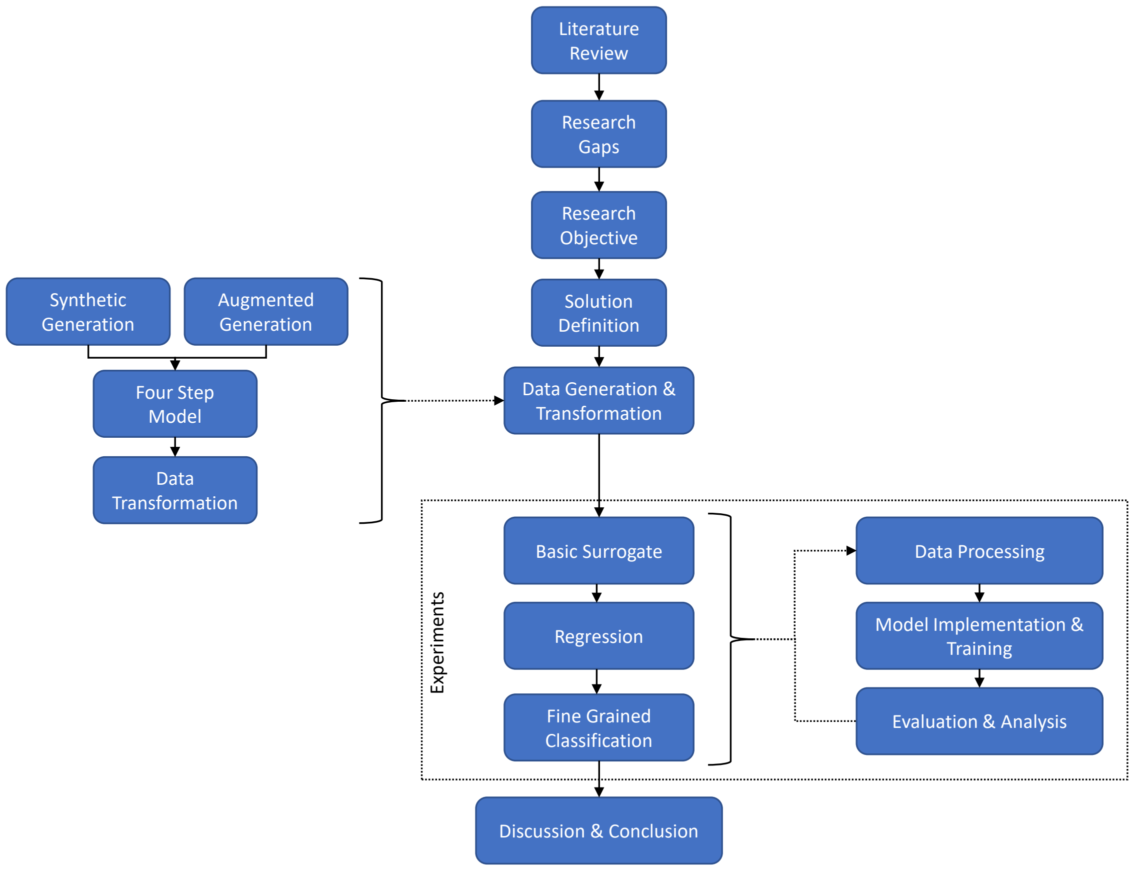


Figure 1 Research framework of the thesis.

1.3. Contributions & Thesis Structure

The primary contributions of the thesis are proposing the surrogate framework to overcome some of the major problems of the Four Step Model. Specifically, two novel methods are created for data generation, followed by using both existing deep graph neural networks and proposing new extensions, the GATv3 and GCN+GAT models. The GATv3 model is the first model to be able to reach large depths and the GCN+GAT model shows an attempt to overcome the over-squashing effect. Additionally, hierarchical regression is proposed to stabilize regression by combining it with classification, as well as a method to convert classification into real values. Through the experiments, an optimal model configuration is found for the current state of the art surrogate model for the Four Step Model, showing a novel application for deep learning approaches. Finally, the key points for future research, based on the experimental results, are to generate more data as well as developing graph neural networks agnostic to network size.

The thesis is structured into nine chapters, which go over the following topics:

- **Chapter 1** introduces the topic, the overall challenges and the research questions.
- **Chapter 2** examines the existing literature in the context of this thesis and looks at the pitfalls of current approaches.
- **Chapter 3** explains the methods behind the surrogate models used in the experiments.
- **Chapter 4** introduces a formal definition for the problem statement and outlines the technical approach to the problem.
- **Chapter 5** introduces the data generation algorithms as well as the transformation pipeline.
- **Chapter 6** presents the experiment setups, iteratively describing the processes and which steps were taken.
- **Chapter 7** describes the experimental results, highlighting the relevant findings.
- **Chapter 8** shows a discussion of the results and brings them into the general context of both deep learning and transport planning.
- **Chapter 9** summarizes the main findings, highlights the contributions, concludes the thesis and presents potential future research directions.

The references, as well as appendix, are at the end of the thesis.

2. Literature Review

The following chapter presents the existing work done relevant to the motivation of this thesis and the direction of research pursued. The chapter does not focus the methods being used in this thesis, which will be examined closely in Chapters 3 & 5, as these only become relevant after finding the literature gaps.

First, the Four Step Model is presented, together with its shortcomings and attempted solutions to solve them. Then, a brief look is taken at end to end deep learning in transport planning, and finally examining the surrogate modelling field with a summary of gaps in the literature.

2.1. Strategic Transport Planning & Four Step Model

There are many subfields to transport planning, but the focus of the thesis is on strategic travel modelling, where it is observed how long term changes in a region affect various transport related factors (Mladenovic and Trifunovic 2014). The specific area of this thesis is examining the effect of socio-economic data together with network information on traffic volumes. Many other aspects also fall into travel modelling but have been exempted for simplicity.

2.1.1. Four Step Model

The Four Step Model is the central approach for travel modelling and has been established through decades of being applied to various scenarios around the world (Mladenovic and Trifunovic 2014; Manheim 1979). The method assumes that the network input is dissected into traffic analysis zones (TAZ), which contain relatively homogeneous socio-economic factors and geographically close land usage objects (e.g. housing) (McNally and Rindt 2007b). The Four Step Model solves the strategic transport problem by applying four steps consecutively, as following:

1. **Trip Generation** - For each zone, a prediction is made as to how many trips start and end there.
2. **Trip Distribution** - Based on various factors, for every zone, it is determined which of the previously generated trips reach which other zones, producing origin-destination trip volumes.
3. **Mode Split** - For each origin-destination trip volume, a distribution across the transport modes is determined.
4. **Traffic Assignment** - Modal trips from each origin-destination pair are assigned to the network, usually aiming at a user equilibrium.

The Four Step Model assumes a hierarchical and independent calculation of the individual steps, however, it can be seen that there are interaction effects. For example, the destination of a person, determined in Stage 2, often depends on the traffic situation, which in itself is dependent on Stage 4. To this end, various extensions have been proposed, which introduce interdependencies and iteratively perform the four steps until a certain convergence criteria is met (Mladenovic and Trifunovic 2014; Boyce, Y.-F. Zhang, and Lupa 1994; Feng, Mao, and Q. Sun 2010; D. Li et al. 2012; Dios Ortúzar and Willumsen 2011; Ali Safwat and Magnanti 1988).

The models for the individual steps are estimated independently based on the provided data and then further calibrated together.

2.1.2. Downsides of the Four Step Model

As with most popular methods, a number of downsides have been identified in existing literature. The downsides here focus on the general model, and not on the individual steps.

Mladenovic and Trifunovic 2014 perform an extensive review summarizing the shortcomings of the Four Step Model. A major concern shown in the review paper is that the model assumes a sequential decision making process, which is not supported by how humans perform decisions for travelling. The authors also show that the approach is deterministic, translating into little alternatives being explored in practise. Furthermore, the paper identifies that the Four Step Model requires an iterative procedure, which becomes computationally intensive on large scale realistic networks. The approach to the prediction cycle is also criticized, as models directly extrapolate on data without analyzing whether policies change the underlying assumptions. The authors find that the data used is often highly aggregated, and not precise enough to calibrate transport models well. The paper also identifies further general issues, such as a lack of trip chaining, poor modelling of induced demand and poor future planning assessment.

Looking more closely at the calibration procedure, Najmi et al. 2020 explains that the process is often done with a major manual component relying heavily on expert knowledge. The authors argue that due to the long run time especially of the traffic assignment step, calibration is usually performed by calibrating the individual steps, thus resulting in a sub-optimal model. Finally, the authors identify that calibration has received less academic interest in comparison to other parts of the modelling process.

Examining the runtime of the Four Step Model more closely, it is considered to be good in comparison to Activity-Based models. However, when looking at absolute numbers, a single run on a large sized region may take a few hours (PTV Group 2021). By enabling a faster runtime it would be possible not only to perform realtime traffic planning by the end user, but also to introduce more automated explorative methods. The runtime has not been discussed much in academic literature, but is an essential component of practical transport planning.

Scope of Literature Review

The aim of the thesis is going to be on tackling the shortcomings of the Four Step Model. Specifically, the focus is on the problems of assuming that travelers make sequential decisions, the model is deterministic, calibration requires manual work and there are long run times of large models. These were selected as the author believe that, from a methodological point of view, they are linked together closely and represent the core downsides of using the Four Step Model. In the following sections, the literature review will focus on approaches attempting to tackle these issues. A literature review looking at attempts to solve the other problems of the Four Step Model is outside the scope of this thesis.

2.2. Existing Work Solving the Four Step Model's Problems

There have been a number of work attempting to solve the four previously stated problems of the Four Step Model. In this section, various work made to overcome these issues are presented.

2.2.1. Combining Individual Steps Together

A large number of work has looked at combining individual steps of the Four Step Model together, to improve the basic model assumption on travelers' decision making process as well to require less manual work for calibration.

Freitas et al. 2019 combine the route and mode choice together, by applying a recursive logit technique. Each link with its respective mode is then considered individually in a hierarchical fashion, with the algorithm moving forward to the connected nodes after a choice is made. The approach assumes an underlying linear relationship of the input data and requires a high memory consumption for evaluation on a real world network. The authors present results showing that the approach finds realistic values for the Zurich network.

Vrtic et al. 2007 describe the EVA algorithm, which combines trip generation, trip distribution and mode choice together in a disaggregate manner. Trip generation is performed as a separate step, but with a coupling to the network supply. Trip distribution and mode choice are combined using a logit model and joint modelling. All models here also assume a linear relationship between the input variables. Again, the authors present results of the model on the Swiss network showing that they appear realistic with respect to the data.

Z. Zhou, A. Chen, and Wong 2009 propose modelling all four stages jointly together, based on the random utility framework in a hierarchical logit model. The choices are decomposed into statistically independent decisions, following the four stages of the Four Step Model. Each decomposition is modelled as a logit model, and finally optimized together. Note, this implies that the proposed model still assumes that the decision making process is sequential, but allows joint optimization and calibration. Additionally, the logit models assume linear relationships between the input variables. The paper does not apply the model to any data.

Ali Safwat and Magnanti 1988 propose the Simultaneous Transportation Equilibrium Model (STEM), which models the Four Step Model as a convex optimization problem. Trip generation is based on a linear model, a logit model is used for the trip distribution, whilst modal split is introduced via network routing and traffic assignment is determined by using the Wardrop user equilibrium as a constraint. Altogether, each step assumes various simplifications to enable the problem to be convex and computationally tractable. One of the key simplifications is that all travelers make similar decisions. An extension to the model is introduced in Hasan and Dashti 2007, the Multiclass Simultaneous Transportation Equilibrium Model (MSTEM), which adds multiclass capabilities for various travelers groups. The model is identical to that of STEM, except with separate trip generation and trip distribution functions. Due to the added functions, the convex property cannot be guaranteed, so the problem is reformulated as a variational inequality solved by a diagonalization algorithm. The user decision making models in MSTEM assume linear interactions between the variables, and the sequential assumptions of the Four Step Model is also found here.

Abrahamsson and Lundqvist 1999 combine trip distribution and mode choice together, similarly to the previous papers using a logit model assuming linear interactivity of the input variables. The authors then apply it to the Stockholm network and compare it to other formulations but find it less accurate than the other versions.

Tan et al. 2019 combine trip distribution and traffic assignment steps together. The core principle of the approach is to convert the combined steps into a standard traffic assignment problem by adding new network components.

McNally and Rindt 2007a introduce the Activity-based models. Activity-based models are a completely new framework to the Four Step Model, attempting to solve the underlying problem of individual steps being the core to the decision making process. The underlying assumption is that a person chooses a specific activity, and the trips are then derived from that choice. The class of approaches is outside the scope of this thesis, but the Activity-based models suffer heavily from the other same three problems that the 4SM suffers described earlier.

To summarize, the work done on combining individual steps has been overwhelmingly focused on models which assume linear interactions of the input variables, such as the logit models. The implication is that there are hard limitations on the input data that can be used, as well as an upper bound as to which effects can be reproduced. Additionally, minimal work has been done in combining all four steps. These factors together result in models which do not entirely solve the problems of the Four Step Model and are still primarily not used in practice.

2.2.2. Machine Learning in the Four Step Model

Machine learning methods have the advantage of being able to model highly complex patterns, which becomes relevant in the case of the Four Step Model to improve its expressiveness. Especially in recent years, with the availability of both more data and better methods, machine learning has become a popular topic, often replacing previously used statistical-based methods.

Looking at trip generation, Saleh et al. 2021 apply neural networks for trip production and attraction rates. Xiao et al. 2021 introduce a new method combining convolutional neural networks with recurrent neural networks. Kaewwichian 2019 use machine learning for car ownership modelling based on decision trees and neural networks. The authors of all of the papers find that machine learning approaches outperform the baselines, though sometimes with overfitting problems and challenges in interpretability.

For the trip distribution stage, Pourebrahim et al. 2019 use gravity models, random forests and neural networks on Twitter data to enhance the model precision. Goel and Sinha 2015 use an adaptive neuro-fuzzy inference system to predict trip distribution in Delhi. Kompil and Celik 2013 combine a genetic algorithm with a fuzzy rule-based system and apply it to the region of Istanbul. Pitombo, Souza, and Lindner 2017 develop decision tree algorithms to take disaggregate factors into account. All papers find again that the machine learning models perform better but have problems with constraints and interpretability.

It can be seen that most machine learning work related to the Four Step Model has been done on mode choice. In this paragraph, only key publications are presented, highlighting the methods used. A detailed machine learning mode choice overview from 2019 can be found in Pineda-Jaramillo 2019. Hagenauer and Helbich 2017 compares traditional logit based models with naive bayes, support vector machine, neural networks, gradient boosting trees, random forests and bagging trees. X. Zhao et al. 2020 also compares various machine learning approaches and additionally introduce a new method for computing elasticities, significant for interpretation. Cheng et al. 2019 examines random forests and logit models more closely on data from the city of Nanjing. The authors of the papers consistently find that random forest approaches work best, however, they often have unrealistic coefficient interpretations which require further processing.

In traffic assignment, Grunitzki and Bazzan 2017 develops a reinforcement learning algorithm to attempt to solve the traffic assignment equilibrium. B. Zhou et al. 2020 also use reinforcement learning, but with an updated potential function. There has been comparatively little research done in this area, and all of it has been focused on reinforcement learning. The papers find that the models can outperform the baselines in certain cases but are highly sensitive to input features.

Taking all four steps into consideration, X. Wu et al. 2018 introduce the Hierarchical Flow

Network. The core idea is to develop a computational graph of all four steps, allowing the computation of a gradient for all steps. In turn, this allows the usage of deep learning methods for all steps and to train the models jointly. The result of such a model is that it allows the traffic modeller to exploit multiple data sources simultaneously. To enable the computation of the gradients, the authors propose to sample the possible paths taken, meaning that not all possibilities are considered. Note, the authors do not introduce a model that combines the steps together, rather to calibrate them simultaneously. Additionally, the method is computationally expensive, growing exponentially with the size of the network.

Summarizing the work of machine learning in relation to the Four Step Model, it can be seen that the majority is done on the individual steps, especially on mode choice. Note that machine learning has not been used to combine individual steps to the best of our knowledge, as described in the previous sub-section.

2.2.3. Uncertainty Quantification in the Four Step Model

By default, the Four Step Model provides deterministic outputs. As highlighted in Chapter 2.1.2, assuming non-deterministic inputs and having a statistical distribution as output would greatly benefit the application of the transport planning. Uncertainty quantification is a field from simulation sciences whereby an input distribution is provided, instead of a single value. It does not directly solve the problem of the deterministic assumptions of the Four Step Model, but is the closest approach found in the existing work. A summary of uncertainty in general transport planning is provided in Navarro-Ligero, Soria-Lara, and Valenzuela-Montes 2019, and the most relevant papers are presented in this subsection which are focused on the entire Four Step Model.

Manzo, O. A. Nielsen, and Prato 2015 uses a Monte Carlo sampling procedure for the Four Step Model on a town in New Zealand, finding that most uncertainty is produced in the first three stages. However, only 100 samples were used, and the number of model parameters assumed uncertain 30. The same authors also analyze the effect of uncertainty in the BPR volume delay function in Manzo, O. A. Nielsen, and Prato 2014, concerning the output of the Four Step Model. In this approach, the authors also use a sampling procedure, namely bootstrapping. Both approaches are highly sensitive to the assumptions of uncertainty search, as they require explicit statistical distributions as input. Additionally, for sampling-based methods, the number of samples needed increases exponentially with the number of uncertain parameters, requiring a long simulation time.

C. Yang et al. 2013 extend the approach shown in Z. Zhou, A. Chen, and Wong 2009 to produce statistical outputs for uncertainty analysis. Here, the authors use logit models for all steps, combining them and producing an output distribution. The underlying logit assumptions limit the application of the model, but the approach is valuable to uncertainty quantification in transport planning.

Summarizing, it can be seen that two general approaches exist: sampling and model simplification. Unfortunately, both approaches have their downsides, which have led to little practical usage of these approaches.

2.2.4. Automatic Calibration

Trying to tackle the problems of manual calibration consuming a significant amount of time for transport modellers, a large amount of work has been done on creating automatic calibration systems. In the following sub-section, a summary of work is provided, with further details which can be found in Najmi et al. 2020; Omrani and Kattan 2012.

Najmi et al. 2020 show that most of the existing literature focuses on-demand model calibration. Demand model calibration aims to adjust the output made by the first three steps without relating it to the underlying network. The problem here is solved chiefly using variational inequality, gradient-based or stochastic methods. The downside of these approaches is that they disregard the relation of demand and supply.

On the other hand, the network calibration, which is mainly needed for the traffic assignment step, is done by focusing on the speed density model using statistical regression approaches (Qin and Mahmassani 2004).

Looking at the combined calibration of all steps, i.e. both demand and supply side, it can be seen that there have been few studies performed, as depicted in Omrani and Kattan 2012. Focusing on recent work, Najmi et al. 2020 propose to use polynomials as a surrogate based on specific input parameters, and then to find their respective optimum. Brinckerhoff 2010 show the calibration to the large scale New York model, using various techniques in a relatively unstructured approach. As described in Chapter 2.2.2, X. Wu et al. 2018 create a sampling based procedure to calibrate all four steps jointly.

Summarizing the approaches to automatic calibration, it can be seen that the majority of automatic calibration procedures aim at optimizing the parameters of a single component of the Four Step Model. The approaches that consider all four stages are susceptible to hyperparameters and require exponential computational time with respect to the number of parameters being optimized. Thus, automatic calibration has had little widespread application.

2.3. End to End Deep Learning for Transport Planning

End to end deep learning has been able to achieve breakthrough results in a number of fields, such as automatic translation and image recognition, by being able to merge multiple modelling steps together into a single large model, such as feature engineering as well as output processing (Pouyanfar et al. 2019; Bahar, Makarov, Zeyer, et al. 2020). In transport planning, the closest work that has been found is in the field of short term traffic prediction, where several work has been recently published. The background here is to apply models in the operational setting. The problem setting is to predict the future traffic conditions, given a current traffic state. The time range considered is usually within a few minutes or hours. There are further sub-problems, such as predicting traffic flow, speed and travel time. An in-depth survey of deep learning in short term traffic prediction is provided by Manibardo, Laña, and Ser 2021, with the most relevant papers and results being summarized in this section. Note, as the thesis is focused on long term transport planning, the following section focuses on the methods and not on the application.

The recent trend has been towards representing the traffic state as a graph. The usual configuration is to have the nodes representing intersections and edges representing street links. With this approach, it is possible to apply graph neural networks, which are more closely examined in the following chapters. There exist alternative formulations, especially to incorporate multiple time steps. With graph neural networks, the state of the art performance on prediction is reached.

Focusing on graph neural network literature, W. Jiang and Luo 2021 provide a survey of the existing research on the topic. There are various specific graph neural network types, and the survey finds that the Graph Convolutional Network (GCN) is the most popular method, followed by the Graph Attention Network (GAT). Further methods were found to be in significantly lower quantities. Additionally, some papers propose extensions, for example, adding a recurrent component to model time dependency. Details on how the models work are presented in further chapters.

The data used for these models are mostly based on traffic sensors or taxi data. Looking at how the data is processed, there is a split between the road connection data or the distance between the nodes. As GCN and GAT both employ only node information, there has been work examining the usage of converting the input graph first into a line graph (Harary and Norman 1960; Ramadan et al. 2020; Xiong et al. 2020).

Summarizing the results, graph neural networks achieve the state of the art performance for short term traffic prediction, though there is very little work done for long term predictions.

2.4. Surrogate Modelling

Coming from simulation sciences, surrogate modelling is when an expensive function is approximated by a function, the *surrogate model*, which posses some positive qualities, such as cheap evaluation costs (Sudret, Marelli, and Wiart 2017). Traditionally, this has been done with polynomials, but recently deep learning models have also started to be used as surrogates and have achieved the state of the art results (L. Sun et al. 2020). Surrogate modelling is a broad field, thus the following section provides a summary of approaches but focuses on deep learning surrogates and, more specifically, on graph neural network surrogates. The focus is taken due to the relevancy of the methods used in this thesis.

Shan and G. G. Wang 2009 provides a summary of classical approaches, which are primarily focused on decomposing the models into linear combinations of simpler functions, such as orthogonal polynomials. As deep learning models are cheap to evaluate and handle high dimensional problems well, they became a natural candidate as surrogates (Tripathy and Bilionis 2018). The common design principle is to generate or retrieve a large number of input samples and feed them through the simulation. The resulting simulation results are then used as targets for deep learning models, whilst the inputs stay the same.

Most surrogate modelling methods have focused on the physics simulation field. However, the data can often be formulated better in non-euclidean formats, specifically as a graph. Thus, a few works have been released on using graph neural networks as surrogate models within the last two years.

Bleeg 2020 define a wind farm as a graph, with each node being a single wind turbine. The authors model the aerodynamic interaction effects between the turbines using a graph neural network surrogate, as simulations take a prohibitively long time for accurate results. The authors use 208 wind farm simulations as training data, exploiting both real and synthetic wind farm configurations. The authors find a good correlation between the predictions and the targets, though they note the simulations were focused on a limited scenario that should be expanded in the future.

Pfaff et al. 2020 reach the state of the art results using graph neural networks for mesh simulation. The authors use as input the discretized mesh of a cloth, formulated as a graph. Given a starting mesh and environment state, the problem aims to predict a number of future cloth states. The authors apply it to several various cloth problems, each with 1200 cloth samples. The authors show a speedup of one to two magnitudes compared to the baseline, with outstanding performance on unseen data and even longer timesteps. Again, the number of cloth scenarios here are limited that should be expanded in the future.

Ogoke et al. 2020 create a GCN surrogate for aerofoil drag force prediction, where the aerofoil environment is modelled as a mesh. The authors take 1522 aerofoils that are then applied with simulations to determine target values. The results show that the model achieves highly accurate results but is again limited to the scenarios tested.

The problems of the Four Step Model are similar to the ones that are solved by surrogate models in other fields. However, to the best of the author's knowledge, there has been no existing literature in building a surrogate model for the Four Step Model at the time of writing.

Examining the closest related literature to this, Najmi et al. 2020 introduce a polynomial surrogate for agent-based simulations, focusing on automatic calibration. The authors propose to use it in an online setting, fitting the polynomials to a small number of performed simulations. However, the authors note that the procedure cannot be used for entirely automatic calibration due to precision limitations.

C. Zhang, Osorio, and Flötteröd 2017 propose to use linear equations for the calibration of the demand side, i.e. the first three steps of the four step model. The authors create an iterative method, similar to Bayesian Optimization (Mockus 1989), whereby simulations are performed with a linear model fitted to the data, followed by the analytical solving of the linear model. The procedure is performed until convergence. The approach is motivated for automatic calibration.

It can be seen that there for the Four Step Model, there has not been an explicit focus on building a surrogate model in the existing literature.

2.5. Summary & Thesis Focus

Having seen the large amount of literature on the related topics, the following paragraphs describe a summary, a number of identified gaps and finally the focus of the problems of the Four Step Model within this thesis.

2.5.1. Summary

The literature review focuses on the topics relevant to the thesis. First, several problems with the existing dominating method, the Four Step Model, have been identified over the years and solutions have also been proposed. Regarding the stage modelling issues, logit based models have dominated the combined approaches. Additionally, machine learning methods have been used in individual steps, whilst automatic calibration has been developed in the academic setting. A few methods have been proposed to introduce uncertainty quantification into the transport planning process. Taking the deep learning perspective, there has been a recent development of short-term traffic predictions, where graph neural networks have reached state-of-the-art results. Finally, surrogate modelling from other fields has shown that it can relieve many of the problems facing simulations by using deep learning.

However, there are a number of gaps found in the literature:

- Methods improving the Four Step Model mostly focus on a single aspect, often neglecting other factors, making it impractical.
- Few methods look at all stages of the Four Step Model, and specifically no methods were found using machine learning to combine methods, which could model non-linear interactions of the steps.
- Little academic interest has been shown in the run time of the models, even though in

practise it is an important component.

- Uncertainty quantification for transport planning has had little research.
- End to end deep learning models have focused solely on short term traffic prediction.
- To the best of our knowledge, no surrogate model has been built for the Four Step Model.

2.5.2. Thesis Focus

Summarizing, the focus of the thesis is on tackling four specific problems of the Four Step Model. These problems are:

1. The need for the assumption that traveler's decision making is done in individual steps, i.e. braking down into smaller, independent models,
2. The underlying deterministic assumptions,
3. Calibration requiring manual work,
4. Long run times of large models.

A visualization is found in Figure 2, which highlights the lack of literature in the area of the proposed method.

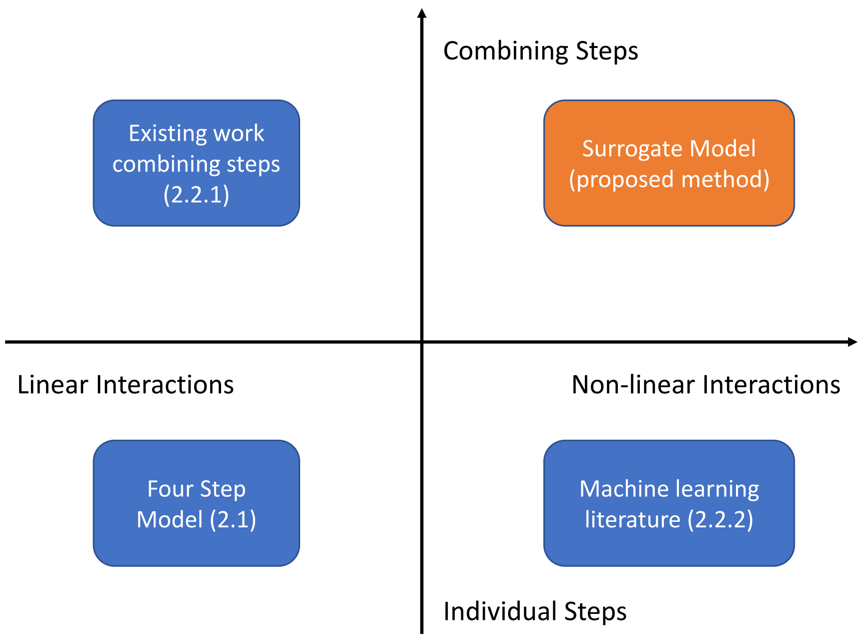

Combining Steps
Existing work combining steps (2.2.1)
Surrogate Model (proposed method)
Linear Interactions
Non-linear Interactions
Four Step Model (2.1)
Machine learning literature (2.2.2)
Individual Steps

Figure 2 A simplified comparison of the proposed method in this thesis to existing literature, from the model expressiveness point of view. The numbers in the parentheses refer to the chapter where more details are found, with blue colours signifying existing fields of research and orange novel fields.

In the following chapters, a method is proposed to tackle these issues, inspired by the work done in the field of surrogate modelling.

3. Surrogate Models

The following chapter examines various surrogate models that are then used in the experimental setup in Chapter 6. In general, any machine learning model can serve as a surrogate, however the focus of thesis is on graph neural networks. Therefore, a collection of relevant methods are elaborated in the following chapter, from which models are composed for various experiments later on.

The chapter starts with the general training procedure, moving on to baseline models, followed by deep learning and then more specifically graph neural networks. Finally, regularization methods, loss functions and evaluation metrics are presented, finishing with the model implementation.

3.1. General Training Procedure, Classification & Regression

Starting with the general training procedure, the input to the models is the network together with the socio-economic data, whilst the targets are the outputs of the Four Step Model. The targets need to be predicted for every link in the network. Thus, the training procedure is a standard supervised learning setup. The training procedure iteratively adjusts the model's parameters to minimize a certain error (Friedman 2017). The exact way that a specific model is trained is dependent on its implementation, and the problem setting. Details on the surrogate problem are found in Chapter 4.

There are two types of problems experimented in this thesis: regression and classification. In the regression setting, the aim is to predict a real value representing transport usage along the specific link. On the other hand, in classification, the aim is to predict a bucket index within which the transport usage lies, with the buckets being defined explicitly by the modeler. The exact motivations and context for each problem is dependent on the aim, and is discussed in the results chapter. The following sections look at the individual models examined in this thesis.

3.2. Baseline Models

First, the baseline models are presented. By performing multiple baselines, it is possible to estimate both how hard a problem is, as well as validate that the more complex models are showing true improvements. The baseline models are taken as those used often in machine learning on a wide variety of tasks. Additionally, baseline models have implementations in existing libraries, making them easy to apply to new data settings.

3.2.1. Majority Classifier

For classification tasks, the majority classifier predicts for all inputs the class that was most often in the training data. This can be seen as the most simple classifier, and is used as the absolute lower bound.

3.2.2. Mean Regressor

For regression, the lower bound model is the mean regressor, which predicts the mean of the training data for all input data. This is the simplest regressor possible, equivalent to a linear regressor which has only the intercept.

3.2.3. Random Forests

The random forest model (Ho 1995) is a machine learning method that is suitable for high variance data. It works by creating and training many decision trees, then averaging their results. The core idea is to overcome overfitting of decision trees by training each one on a subsample of the data. The exact training sequence is outside the scope of this thesis, but in summary, the algorithm works by comparing for each decision tree, for each branch split, whether it would improve the Gini impurity of the tree (D'Ambrosio and Tutore 2011). The practical result of random forests is that they are quick to train and require little parameter tuning, meaning that they are often used as a good baseline for machine learning methods. The random forest is implemented in Scikit-Learn (Pedregosa et al. 2011), using default parameters if not stated otherwise.

3.3. Notation

Before diving into the formal details of the surrogate models, the following table contains all symbols and terms used within this thesis, shown in their order of appearance.

Symbol	**Type**	**Explanation**
		Surrogate Models
h_i^j	$\mathbb{R}$	Output of neuron i in layer j
w_{ik}^j	$\mathbb{R}$	Weight of neuron i in layer j for neuron k in layer $j-1$
s_j	$\mathbb{N}_{>0}$	Number of neurons in layer j
K	$\mathbb{N}_{>0}$	Number of layers
$\sigma(x)$	Function	Activation function run on input x, element-wise

$\sigma_{\mathit{tanh}}(x)$	Function	Activation function of the hyperbolic tangent
$\sigma_{\mathit{relu}}(x)$	Function	Activation function of the relu function
$\boldsymbol{h}^j$	$\mathbb{R}^{s_j}$	Vector output of all neurons in layer j
$\boldsymbol{W}^j$	$\mathbb{R}^{s_{j-1} \times s_j}$	Weight matrix of layer j
$\boldsymbol{h}^i$	$\mathbb{R}^{s_j}$	Temporary vector used for GNN computations for node i
$\boldsymbol{n}_i^j$	$\mathbb{R}^{s_j}$	Output of node i in layer j of a GNN
$n(i)$	Function	Returns the set of neighbours of node i
$N_{ff}(\boldsymbol{x})$	Function	Single feed forward layer, performed on input vector $\boldsymbol{x}$
$[;]$	Function	Concatenates all input vectors or matrices along the first axis together
α_{ik}	$\mathbb{R}$	Attention weight of node i for neighbour node k
α	$[0,1]$	Hyperparameter to set average weighting, used in GATv3 and GCNII models
$\sigma_{softmax}(x)_k$	Function	Softmax activation function applied to input x, and taking the output for node k
$f(x)$	Function	General trained machine learning model taking x as input and producing an output.
$\mathbb{1}_{(x)}$	Function	Indicator function, which is 1 when statement x is true, else 0
$MAE_{\geq 10}$	$\mathbb{R}_{>0}$	Mean Absolute Error, evaluated only on samples where targets have at least 10 respective units per hours
$R^2_{\geq 10}$	$(-\inf, 1]$	R^2, evaluated only on samples where targets have at least 10 respective units per hours

Problem Formalization

$\emptyset$	Set	Empty set
$\mathcal{A}$	Set	Set of all links in network, together with their respective properties, including capacities and PuT lines
$\mathcal{N}$	Set	Set of all nodes in network
$G(\mathcal{N}, \mathcal{A})$	Tuple	Network data, containing all transport supply information
$\mathcal{S}$	Set	Set of socio-economic data, relating to a network
$\mathcal{T}$	Set	Set of resulting traffic flow, relating to a given $G(\mathcal{N}, \mathcal{A})$ and $\mathcal{S}$
$\mathcal{S}'$	Set	Set of all socio-econimic data states, $\mathcal{S} \in \mathcal{S}'$
$\mathcal{N}'$	Set	Set of all network data states, $G(\mathcal{N}, \mathcal{A}) \in \mathcal{N}'$
$\mathcal{T}'$	Set	Set of all traffic data states, $\mathcal{T} \in \mathcal{T}'$
$\mathcal{D}$	Set	Observed data consisting of triples of $(\mathcal{S},\ G(\mathcal{N}, \mathcal{A}),\ \mathcal{T})$ from a single observation
$\mathcal{D}_{\textit{Synthetic}}$	Set	Synthetic data consisting of triples of $(\mathcal{S},\ G(\mathcal{N}, \mathcal{A}),\ \mathcal{T})$ from a single observation, with the data created by a generator
$f_{TP}(\mathcal{S}, G(\mathcal{N}, \mathcal{A}))$	Function	General transport planning model, takes $\mathcal{S}, G(\mathcal{N}, \mathcal{A})$ as input and produces a predicted traffic flow state
$f_{4SM}(\mathcal{S}, G(\mathcal{N}, \mathcal{A})$	Function	Calibrated Four Step Model to observed data, taking $\mathcal{S}, G(\mathcal{N}, \mathcal{A})$ as input and producing a predicted traffic flow state
$g(x)$	Function	General surrogate model to run on input x
$\mathcal{G}$	Set	Set of all potential surrogates for the Four Step Model

$\mathcal{G}_{GNN}$	Set	Set of all GNN functions
$e_{\mathcal{D}}(f, T)$	Function, $[0, \inf)$	Error function over a given dataset and a model function to evaluate, with 0 meaning perfect predictions
$\mathbb{G}(f_{4SM})$	Function	Generator that with every call creates a, potentially random, network with socio-economic data, runs the f_{4SM} on it and returns the results as a triple of $(\mathcal{S},\ G(\mathcal{N}, \mathcal{A}),\ \mathcal{T})$
$\mathbb{G}_m$	$\mathbb{N}_{>0}$	Number of samples to generate for the synthetic or augmented dataset
$t(g, \mathcal{G}_{GNN}, \mathcal{D})$	Function	Training function to optimize a model to minimize the respective error over the dataset, with the model being from the class $\mathcal{G}_{GNN}$ and initialized with the parameters of g
	Data Generation	
$\mathbb{G}_{nodes}$	$\mathbb{N}_{>0}$	Number of nodes to generate
$\mathbb{G}_{zones}$	$\mathbb{N}_{>0}$	Number of zones to generate
$\mathbb{G}_{zones/nodes}$	$\mathbb{R}_{>0}$	Number of zones to create for each node, set to 0.1 in this thesis
$\mathbb{G}_{buslines/zones}$	$\mathbb{R}_{>0}$	Number of bus lines to create for each zone, set to 1.0 in this thesis
	Experiments and Results	
$N_{embedding}(i)$	Function	Embedding operator that takes a class id i as input and retrieves a differentiable, fixed sized vector from memory, representing that class in a learnt vector space
$\mathbb{E}(X)$	Function	Expectation over a specified distribution over random variable X
$b_{k,1}$	$\mathbb{R}$	Lower bound of bucket k
$b_{k,2}$	$\mathbb{R}$	Upper bound of bucket k

3.4. Deep Learning

Having seen the baselines, the next methods are the most flexible found in machine learning (Hornik, Stinchcombe, and White 1989): deep learning. Deep learning, also sometimes called neural networks, is a type of machine learning approaches that due to their vast flexibility have achieved state of the art results in many complex fields, as shown in the literature review. In this thesis a basic deep learning model is used as a baseline surrogate, whilst more advanced, graph specific deep learning methods are used to get the best surrogate function.

3.4.1. Basic Functionality

The basic functionality of deep learning (Friedman 2017) is based on the fundamental element: the neuron. Each neuron works by taking the input, multiplying it with a learnable weight, and then summing all the resulting values together. The resulting value is then passed through a non-linear, differentiable function called the activation function. The final resulting value is then considered the output of the neuron.

Neurons are stacked together in layers with a predetermined and fixed number of neurons and layers. Each layer takes the output of the previous layer as input, and then produces its own output based on that. The first layer is the input, and the final layer is the output of the model, whilst the intermediate layers are the hidden layers. Deep learning models are those considered to have more than one hidden layer and the entire construct is called an artificial neural network. Formally, a neuron can be expressed as

$$h_i^j = \sigma\left(\sum_{k=1}^{s_{j-1}} h_k^{j-1} \cdot w_{ik}^j\right), \tag{3.1}$$

with h_i^j being the output of neuron i in layer j, σ being the activation function, w_{ik}^j the weight of neuron i in layer j for the neuron k in layer $j-1$, s_{j-1} the number of neurons in layer $j-1$ and h_k^{j-1} the output of neuron k in layer $j-1$. A visualisation is provided in Figure 3.

Due to its easy computation and good performance, the primary activation function used throughout this thesis is the relu function (Shang et al. 2016), which is defined as follows:

$$\sigma_{\textit{relu}}(x) = max(0, x). \tag{3.2}$$

The hyperbolic tangent function (tanh) is used in some of the experiments, e.g. hierarchical regression of Chapter 6.2.2, which squashes the input into the range of -1 to 1. The definition is as following:

$$\sigma_{\textit{tanh}}(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \tag{3.3}$$

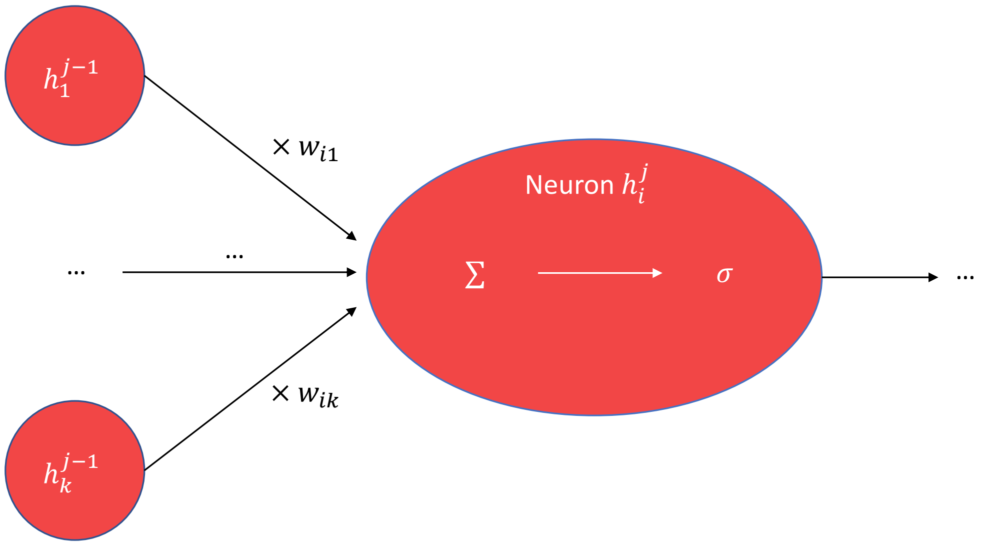


Figure 3 A visualisation of how a neuron works.

Note that the operations for a neural network layer can easily be expressed in matrix multiplication, which leads to the extensive speedup found when performing the calculations on GPUs:

$$\boldsymbol{h}^j = \sigma\left(\boldsymbol{h}^{j-1} \times \boldsymbol{W}^j\right). \tag{3.4}$$

In this formulation the operations from Eq. 3.1 are performed simultaneously for all neurons in layer j. $\boldsymbol{h}^j$ is the vector containing the outputs of layer j, $\times$ is the matrix-vector multiplication, and $\boldsymbol{W}^j$ is the weight matrix. The activation function is then applied element-wise on the vector. Often such a neural network layer is called a feed forward layer.

3.4.2. Deep Learning Training

In order for the models to be useful and output realistic predictions, the weight matrices need to be tuned, using a training procedure. The following paragraphs provide a high level overview of how training works for neural networks.

It can be seen that each operation previously defined is differentiable. Thus if a differentiable cost function is used, it is possible to recursively apply the chain rule from calculus to obtain the gradient of each weight with respect to the cost function, given a specific sample. This procedure is called backpropagation (Rumelhart, Hinton, and Williams 1986). Given the gradients, it is possible to optimize the weights to minimize the cost function using gradient descent. Note, there is no guarantee of convergence for deep learning.

Initially, the entire training dataset needed to be evaluated before the final gradient was calculated. However, to be more computationally efficient stochastic gradient (Kiefer and Wolfowitz

1952) descent is employed, which updates the weights every few samples. The stochastic nature helps the model to reduce the number of computations needed.

Gradient descent is also prone to get stuck in local minima. To overcome this, the Adam optimizer was introduced, which adds momentum as well as some other minor tricks to gradient descent, making it more stable (Kingma and Ba 2015). The formal definitions for deep learning training can be found in (M. A. Nielsen 2015), however they are outside the scope of this thesis.

3.4.3. Deep Learning Baseline

The deep learning model, as described before, is also called the Multilayer Perceptron (MLP). Even though further advanced models are used, an MLP is also used as a baseline model in experiments. If not otherwise stated, the MLP contains three hidden layers, each with 100 neurons, and training is done with the Adam optimizer. Training is performed as long as possible until three consecutive epochs, i.e. steps through the entire dataset, have a loss function difference less than 10^{-4}. The MLP is implemented in Scikit Learn.

3.5. Graph Neural Networks

The MLP is a highly flexible model, however, it does not explicitly consider the graph structure of the transport network in its computations. Here, the class of graph neural networks (GNNs) extend the deep learning framework to directly use graph information. The following section first looks at the basic graph convolutional network, then examines the graph attention network and finally the newly proposed combined model.

3.5.1. Graph Convolutional Neural Networks

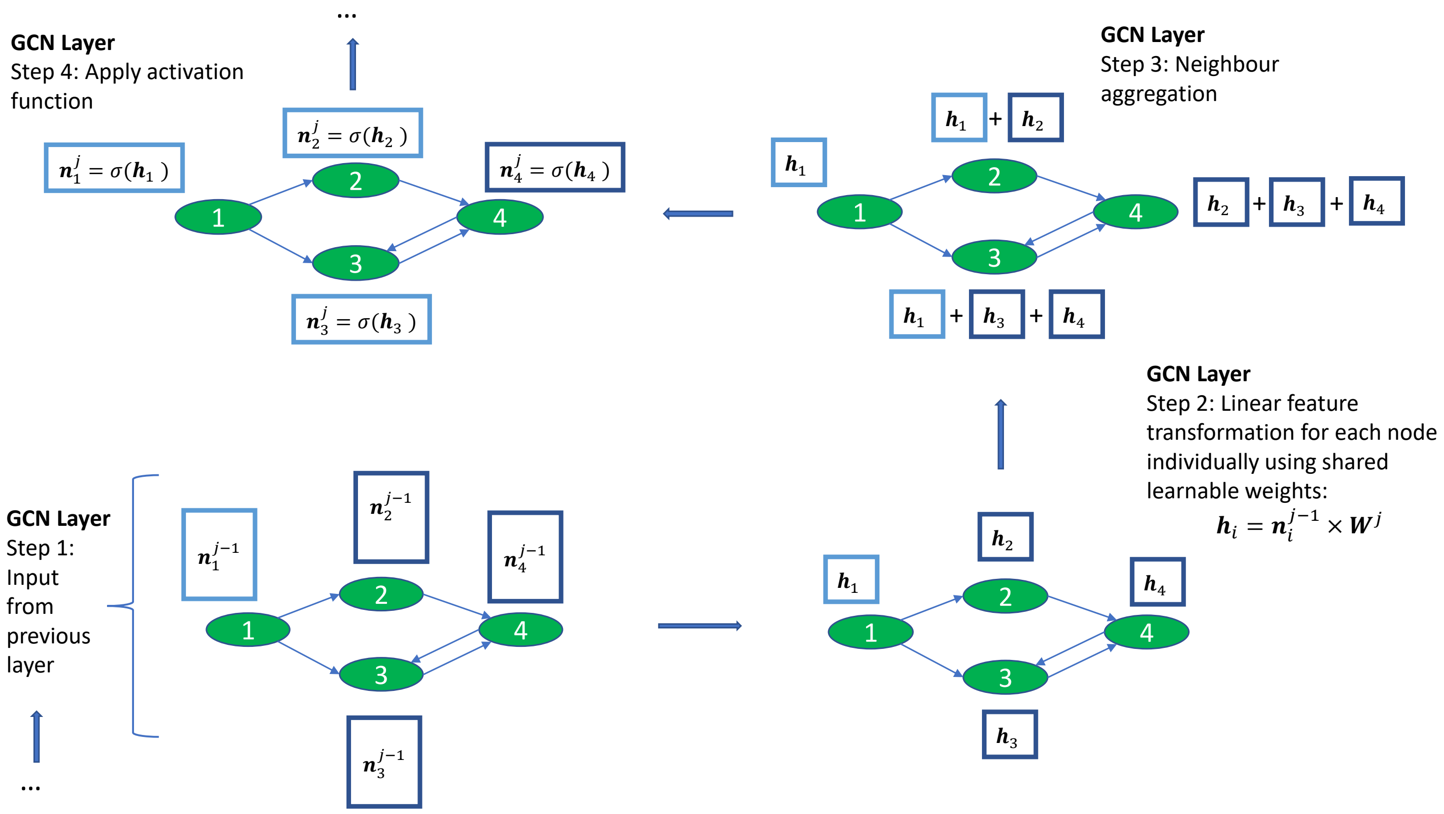
Graph convolutional neural networks (GCN) (Kipf and Welling 2017) was the first graph GNN to be introduced. A GCN is a neural network that consists of stacked GCN layers. The input of a GCN is the ordered nodes and their features. Then, in each GCN layer, the neighbouring nodes exchange their feature vectors before performing a standard MLP step. The output of a GCN is thus also on the same graph as the input data, but with processed features, such as a bucket classification.

The intuition of a GCN with n layers is that any two nodes within n hops over the graph will exchange their information. Thus, the number of layers selected for the entire network also determines the maximum graph size that can be effectively used for transport planning. Concretely, the following steps are applied in each GCN layer.

1. First, the output of the previous layer is taken. If this is the first GCN layer, then the input is taken. Note, each node of the network has a feature vector, meaning each node is processed by the network.

2. Next, a linear transformation is applied to each node using a shared weight matrix.
3. For each node, take the transformed feature vectors of their incoming neighbours and add them onto their own feature vector.
4. Apply the activation function on all nodes and features element-wise.

An illustration is provided in Figure 4.



GCN Layer
Step 4: Apply activation function
$n_1^j = \sigma(h_1)$
$n_2^j = \sigma(h_2)$
$n_3^j = \sigma(h_3)$
$n_4^j = \sigma(h_4)$
GCN Layer
Step 3: Neighbour aggregation
h_1
$h_1 + h_2$
$h_2 + h_3 + h_4$
$h_1 + h_3 + h_4$
GCN Layer
Step 2: Linear feature transformation for each node individually using shared learnable weights:
$h_i = n_i^{j-1} \times W^j$
GCN Layer
Step 1: Input from previous layer
n_1^{j-1}
n_2^{j-1}
n_3^{j-1}
n_4^{j-1}

Figure 4 The inner workings of a single GCN layer. The lighter coloured boxes around the vectors show at each stage where the information of node 1 has reached.

The steps can also be formally expressed as,

$$n_i^j = \sigma\left(\sum_{k\in n(i)\cup\{i\}} n_k^{j-1} \times W^j\right), \tag{3.5}$$

with n_i^j being the feature of node i in GCN layer j, σ the activation function, k all direct neighbours of i and W^j the weight matrix of layer j. Note that in a single GCN layer, the weight matrix W^j is shared. Regarding the training of the GCN, each operation is differentiable, meaning that backpropagation can be employed to calculate the gradient of all weights. Finally, stochastic gradient descent is applied to optimize the weights using the Adam optimizer.

3.5.2. GCNII

A problem with the GCN formulation presented above is that it suffers from the over-smoothing problem (Q. Li, Han, and X.-M. Wu 2018) and over-squashing (Alon and Yahav 2020). The over-smoothing problem is when the features of the individual nodes become highly similar to each other as more GCN layers are stacked together. To overcome the over-smoothing issue, (M. Chen et al. 2020) propose two extensions. The first extension is a residual connection to the initial layer, meaning that a weighted average is performed for each node at every layer between its current feature and the input. The second extension is to perform a weighted average between the shared weight matrix and the identity matrix. The averaging weights, defined as α and θ respectively, are hyperparameters of the model. The result of these two extensions is that the model can have over 60 layers without performance degradation. For the formal definitions, the reader is referred to the original article. The modified model is named as GCNII. The hyperparameters are analyzed and discussed in the experiments chapter later in the thesis.

The over-squashing issue is the problem that for a given graph, the amount of information each node needs to compress into a single fixed sized feature vector grows exponentially with the number of layers. The result is that information is lost when many layers are stacked. This issue is not solved with GCNII and is discussed in later sections.

Summarizing the GCN section, the core addition is that by using local neighbour information exchange, it is possible to extend deep learning models to explicitly use graph structure information. With the extensions proposed in GCNII, the models overcome numerical problems and are able to be trained in deep models. Further models now attempt to improve the modelling capabilities.

3.5.3. Graph Attention Networks

It can be seen that the neighbour aggregation function in the GCN puts equal weight on all neighbours. However, not all neighbours are equally relevant. An example would be where at a junction, two roads come in and one goes out. One incoming road has a high traffic demand, whilst the other has zero. Thus, the information relevant for the outgoing road comes more from the high demand incoming road. Therefore, in this section, the graph attention networks are presented, together with a novel extension to enable them to generalize to deeper networks.

To be able to understand the difference between different neighbours, the concept of attention was introduced to form the Graph Attention Networks (GAT) (Brody, Alon, and Yahav 2021). Attention is determined by calculating a set of weights for each neighbour of a node, with every weight being between 0 and 1, and all the weights summing up to 1. When summing up the transformed feature vectors of a node's neighbours, the vectors are multiplied by the respective weights. The weights themselves are calculated by applying a one layer MLP, which takes in both the original node's and its neighbour's feature vector as input. The steps become as follows:

1. First, the output of the previous layer is taken. If this is the first GAT layer, then the input is taken. Note, each node of the network has a feature vector.
2. Secondly, the attention weights are calculated for each neighbour of a node by a one layer MLP.
3. Next, a linear transformation is applied to each node using a shared weight matrix.
4. For each node, take the transformed feature vectors of their incoming neighbours, multiply the vector by the respective attention weight and add it onto their own feature vector.
5. Apply the activation function on all nodes and features element-wise.

Formally, for each node the following calculations are performed,

$$\begin{aligned} \boldsymbol{n}_i^j &= \sigma\left(\sum_{k\in n(i)\cup\{i\}} \alpha_{ik}\cdot \boldsymbol{n}_k^{j-1}\times \boldsymbol{W}^j\right), \\ \alpha_{ik} &= \sigma_{softmax}\left(N_{ff}\left(\left[\boldsymbol{n}_k^{j-1};\boldsymbol{n}_i^{j-1}\right]\right)\right)_k, \\ &\text{with } \sum_k \alpha_{ik}=1, \text{ and } \alpha_{ik}\in[0,1]. \end{aligned} \tag{3.6}$$

Again, $\boldsymbol{n}_i^j$ is the feature of node i in GAT layer j, σ the activation function, k all direct neighbours of i and $\boldsymbol{W}^j$ the weight matrix of layer j. α_{ik} is the attention weight of node i for neighbour k, with the attention weights summing up to 1 for all of a given nodes neighbours, and each weight being between 0 and 1. N_{ff} is a one layer MLP as defined above. The $[;]$ operator concatenates the input vectors. The activation function $\sigma_{softmax}$ forces all outputs to be between 0 and 1, and normalizes all outputs so that they sum to 1 (Nwankpa et al. 2021). Formally, for output k:

$$\sigma_{softmax}(\boldsymbol{x})_k = \frac{e^{\boldsymbol{x}_k}}{\sum_l e^{\boldsymbol{x}_l}}. \tag{3.7}$$

Please note that the version presented here is GATv2, with the original GAT publication having some minor mathematical problems (Veličković et al. 2018; Brody, Alon, and Yahav 2021). A visualisation is shown in Figure 5. Additionally, often multiple attentions are performed in parallel and their outputs either concatenated or averaged, also called attention heads. By performing multiple attention calculations in parallel, it has been found to be both more

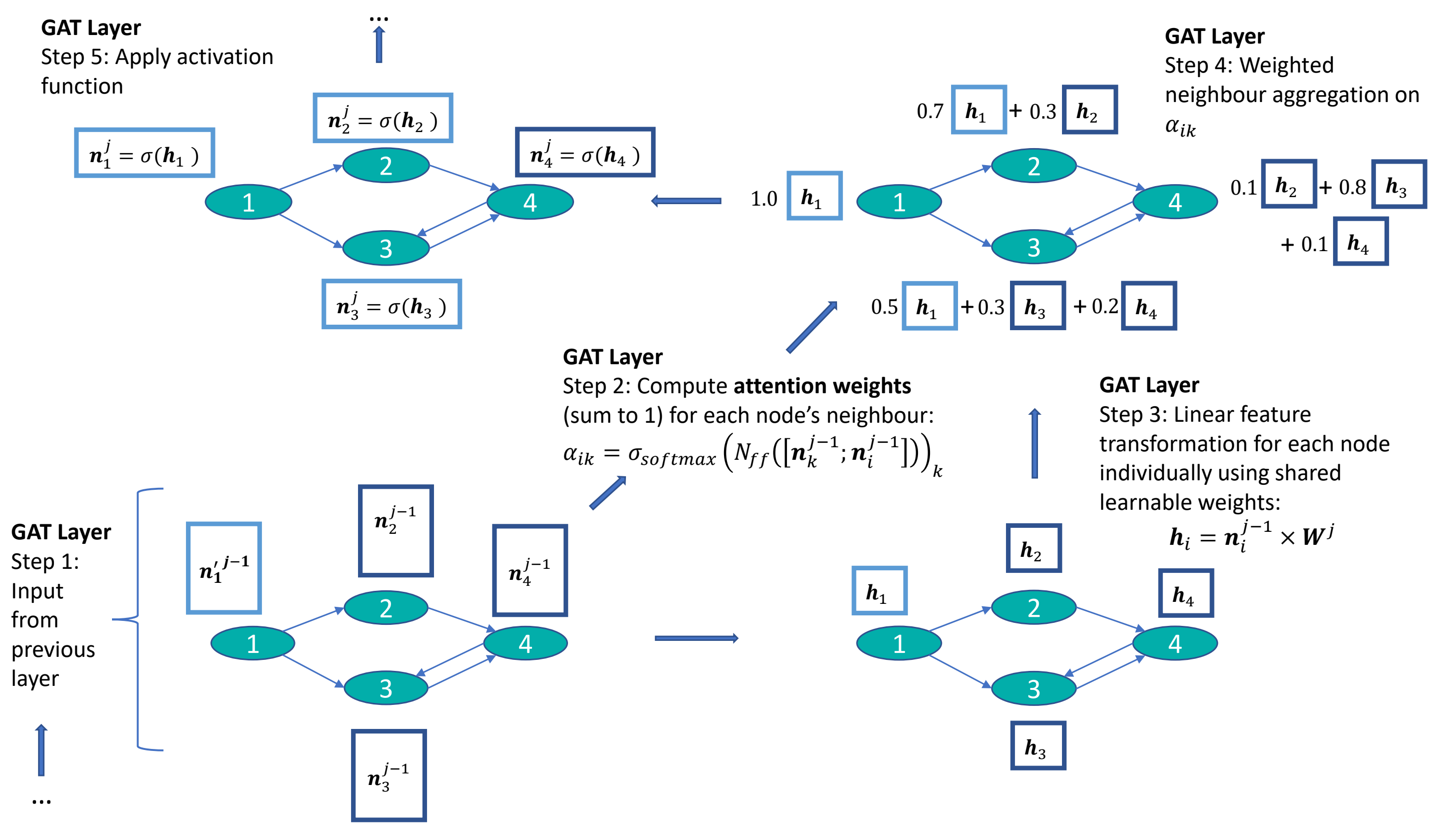

GAT Layer
Step 5: Apply activation function
$n_1^j = \sigma(h_1)$
$n_2^j = \sigma(h_2)$
$n_3^j = \sigma(h_3)$
$n_4^j = \sigma(h_4)$
GAT Layer
Step 4: Weighted neighbour aggregation on α_{ik}
$0.7\ h_1 + 0.3\ h_2$
$1.0\ h_1$
$0.1\ h_2 + 0.8\ h_3 + 0.1\ h_4$
$0.5\ h_1 + 0.3\ h_3 + 0.2\ h_4$
GAT Layer
Step 2: Compute attention weights (sum to 1) for each node's neighbour:
$\alpha_{ik} = \sigma_{softmax}\left(N_{ff}([n_k^{j-1}; n_i^{j-1}])\right)_k$
GAT Layer
Step 3: Linear feature transformation for each node individually using shared learnable weights:
$h_i = n_i^{j-1} \times W^j$
GAT Layer
Step 1: Input from previous layer
$n_1'^{j-1}$
n_2^{j-1}
n_3^{j-1}
n_4^{j-1}
h_1 h_2 h_3 h_4
1 2 3 4

Figure 5 The inner workings of a single GAT layer. The lighter coloured boxes around the vectors show at each stage where the information of node 1 has reached.

expressive and stable during training (Vaswani et al. 2017). Within the scope of this thesis, attention heads are not analysed deeply, due to performance limitations of the experiment machine.

3.5.4. GATv3

Both GAT and GATv2 still suffer from the over-smoothing problem, much like the original GCN. This results in both models performing poorly when trying to stack more than 5 layers. To the best of the author's knowledge, there is no solution proposed to this at the time of writing.

Inspired by the results of GCNII, an novel extension is proposed to use the same simple residual trick as developed by (M. Chen et al. 2020). The formulation thus becomes as follows.

$$
\begin{aligned}
\boldsymbol{n}_i^j =& \sigma\left((1-\alpha)\cdot\left(\sum_{k\in n(i)\cup\{i\}}\alpha_{ik}\cdot\boldsymbol{n}_k^{j-1}\times\boldsymbol{W}^j\right)+\alpha\cdot\boldsymbol{n}_i^0\right),\\
&\alpha_{ik} = \sigma_{softmax}\left(N_{ff}\left(\left[\boldsymbol{n}_k^{j-1};\boldsymbol{n}_i^{j-1}\right]\right)\right)_k,\\
&\text{with } \sum_k \alpha_{ik} = 1, \text{ and } \alpha_{ik}\in[0,1].
\end{aligned}
\tag{3.8}
$$

α is a fixed hyperparameter, and $\boldsymbol{n}_i^0$ is from the input layer for node i. Even though it is a minor change, this solves the issue of developing deep GAT models, as shown in later chapters, and from now this model is denoted as GATv3. Due to time constraints, an in-depth analysis of the over-smoothing problem is not performed in this thesis. Note, that due to the extra computations, the number of GAT layers which can fit onto one GPU is significantly less than those of GCNs. The exact details are shown in the Chapter 7.

Looking at the GAT section, an extension is shown to be able to understand the difference between different neighbours. With the GATv3 extension, deeper models are attempted to be built. However, none of the previous models look at the current issues of GNNs performing poorly on larger graphs.

3.5.5. Combined GAT and GCN

The previous methods show how to solve the over-smoothing problem, but not that of over-squashing. Over-squashing is the effect where each additional GNN layer increases exponentially the amount of information that needs to contained within the fixed sized feature vector of a node (Alon and Yahav 2020). The effect is shown to lose information, which is critical for proper modelling. To the best of the author's knowledge, there is no known widespread fix to this problem. The following section introduces a novel hybrid model of the two previously presented GNNs, together with a graph processing extension, to attempt to overcome the issues of over-squashing on large graphs.

The over-squashing problem can potentially pose a major problem for transport planning predictions, as information needs to be exchanged between nodes which are a large distance away from each other. This is especially important for nodes containing zonal information.

To attempt to overcome this possible issue, a combined GAT and GCN model is proposed. The idea is to first have a number of GATv3 layers, using an artificial graph connecting only zone nodes together. A visualisation of the difference between the artificial zone graph and the true graph is shown in Figure 6. Next, the input is fed through a larger number of GCNII layers, which use the entire network graph. Finally, feed forward layers do the last processing of the data and output link level predictions. An illustration of the model setup can be found in Figure 7.

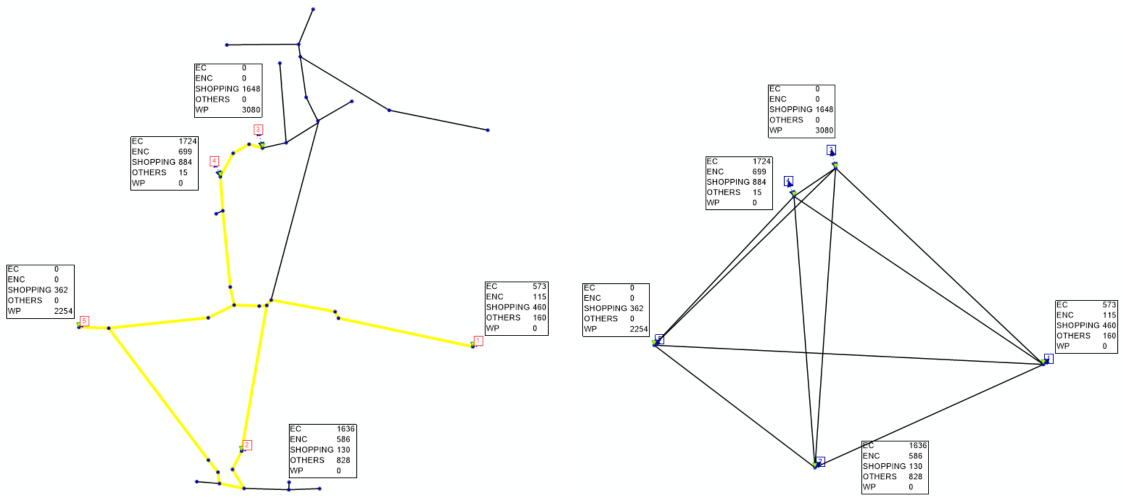

EC 0 ENC 0 SHOPPING 1648 OTHERS 0 WP 3080
EC 1724 ENC 699 SHOPPING 884 OTHERS 15 WP 0
EC 0 ENC 0 SHOPPING 362 OTHERS 0 WP 2254
EC 573 ENC 115 SHOPPING 460 OTHERS 160 WP 0
EC 1636 ENC 586 SHOPPING 130 OTHERS 828 WP 0
EC 0 ENC 0 SHOPPING 1648 OTHERS 0 WP 3080
EC 1724 ENC 699 SHOPPING 884 OTHERS 15 WP 0
EC 0 ENC 0 SHOPPING 362 OTHERS 0 WP 2254
EC 573 ENC 115 SHOPPING 460 OTHERS 160 WP 0
EC 1636 ENC 586 SHOPPING 130 OTHERS 828 WP 0

Figure 6 An example of a true network graph (left) vs zone graph (right), rendered in PTV Visum 2021.

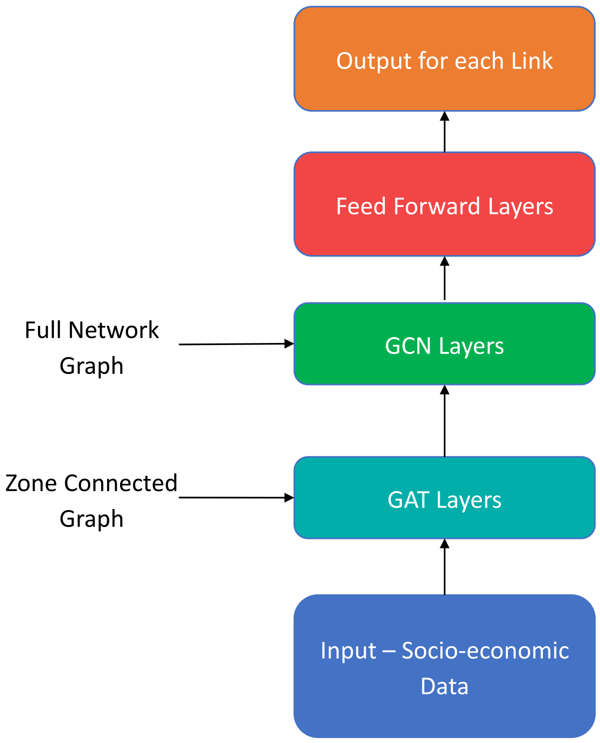


Figure 7 A visualisation of the setup for the combined GCN and GAT model.

The intuition behind this model is similar to that of the Four Step Model in that the first GAT layers focus on determining the demand of the individual zones. The following GCN layers then focus on realistically applying the determined demand onto the existing network. The artificial graph used in the GAT layers is a graph where all zonal nodes are connected, whilst all other nodes are disconnected. This ensures a focus on the socio-economic information governing the demand aspect of transport planning.The model type is called the combined GAT and GCN model, and is abbreviated as GCN + GAT.

The three GNNs presented in this section all share the same basic principle of neighbour information exchange, though with different modifications. An overview is provided in Table 2. By using differentiable functions all of them can be trained with the backpropagation algorithm, and they are the main methods used for surrogate modelling in the experiments. However, the models are incredibly complex, making training often unstable or coming into suboptimal configurations. To overcome these issues, the next section goes into regularization.

Table 2 Comparison of the GNNs used in this thesis.

Model	Advantages	Disadvantages	Literature
GCN	Simple & easy to implement	Cannot make deep models or consider relevancy of neighbours	(Kipf and Welling 2017)
GCNII	GCN, but can be stacked deep, important for this task and easy to implement	Does not consider relevancy of neighbours	(M. Chen et al. 2020)
GATv2	Takes relevancy of neighbours into account	Cannot make deep models	(Brody, Alon, and Yahav 2021)
GATv3	GATv2, but can be stacked deep, important for this task	Requires more GPU memory	Novel
GCN+GAT	Attempts to overcome over-squashing by taking best of GCNII and GATv3	Requires more memory and harder to implement	Novel

3.6. Regularization

Neural networks are usually overparameterized, meaning there are more free parameters that need to be trained than there are data samples (Belkin et al. 2019). The effect often leads to neural networks overfitting on the training dataset, where they memorize the values instead of learning the underlying pattern of the data, thus performing poorly. Regularization techniques attempt to overcome this issue, and specifically two state of the art methods are used extensively in this thesis and are explained in the following paragraphs.

3.6.1. Dropout

Dropout (Srivastava et al. 2014) is a simple regularization technique which randomly drops a pre-determined percentage of connections between two layers. The core idea is to force the layers to learn robust representations which are not dependent on specific outputs. Note, dropout is only applied during training between all layers, whilst during evaluation runs the full network is used.

3.6.2. GraphNorm

GraphNorm (Cai et al. 2021) is a recently proposed method that improves convergence properties of many popular GNNs by standardizing the data between layers. The idea is to standardize each node's feature vector by subtracting the mean of all node's vectors, and then applying a shift by a learned coefficient. Finally, each feature vector is divided by its standard deviation. The authors of the paper show both improvements in convergence and generalization performance for a number of GNNs. When GraphNorm is used, it is applied after a graph operation but before the activation function. Note, GraphNorm is not always used, due to the extra computational effort and GPU memory usage. The experiments that use GraphNorm are clearly defined in the experiment chapter further on.

As shown in later experiment chapters, both regularization techniques are used. Other regularization techniques exist, however dropout and GraphNorm have been shown to be the best in current GNN literature. The final component needed to train GNNs is the loss function, whilst evaluation metrics are used to validate that the model is performing well through different perspectives.

3.7. Loss Functions and Evaluation Metrics

As described in the deep learning section, each neural network requires a loss function with which the gradients of the weights are calculated and finally optimized for training. On the other hand, evaluation metrics summarize the performance of the network from a specific viewpoint and quantitatively validate how well they work. All loss and evaluation functions take a model as well as a dataset as input, whilst providing a certain score as output. However, multiple functions should be considered simultaneously to validate certain improvements, and qualitative analysis should also always be performed. The following section first looks at the loss functions used and then at different evaluation metrics.

3.7.1. Loss Functions

Loss functions are used to train graph neural networks, by providing a score and consequential gradient for a prediction of the model on a dataset. To be differentiable, each component of the loss function needs to be differentiable. Additionally, loss functions need to be numerically stable and fast to evaluate on GPUs. The overall aim of training is thus to create a model which minimizes the loss function.

Mean Squared Error

For regression tasks, i.e. where a precise value is provided as output, mean squared error (MSE) is used as the loss function (Fürnkranz et al. 2011). MSE is defined as follows,

$$MSE = \frac{1}{|dataset|} \sum_{(x,y) \in dataset} (y - f(x))^2 \, , \tag{3.9}$$

with f being a trained model, $dataset$ being a dataset containing input data (x) and respective target data (y). MSE has the advantage of punishing larger errors more than smaller errors, which often leads to more stable results.

Cross Entropy

In classification tasks, a softmax activation function is used, meaning the outputs can be interpreted as probabilities across the classes. In this case, cross entropy (CE) is used as the loss function for training, which is defined as following (Miller, Goodman, and Smyth 1993):

$$CE = -\frac{1}{|dataset|} \sum_{(x,i) \in dataset} log_2(f(x)_i), \tag{3.10}$$

with $dataset$ in this case containing the input x and the respective target class i. Then, $f(x)_i$ is the output probability of the model of class i.

3.7.2. Classification Evaluation Metrics

In classification, a number of metrics are used to determine various performance characteristics. Before, going into further details, four abbreviations are introduced to make formulations easier, assuming a given model and evaluation dataset:

- True Negative (TN) - The amount of correct predictions of a sample not being a specific class,
- False Negative (FN) - The amount of incorrect predictions of a sample not being a specific class,
- False Positive (FP) - The amount of incorrect predictions of a sample being a specific class,
- True Positive (TP) - The amount of correct predictions of a sample being a specific class.

Accuracy

The easiest to interpret evaluation metric is accuracy, defined as following (Friedman 2017).

$$accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{3.11}$$

A perfect classification results in an accuracy of 1, whilst the worst performance has an accuracy of 0. The problem with accuracy is that if a dataset is biased towards a class, it will not compensate for this.

F1 - Score

To improve the bias estimate, the F1 score is introduced (Sasaki et al. 2007).

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{3.12}$$

Here perfect classification is achieved at 1 and the poorest classification at 0. However, even if a dataset is biased, the F1 score only improves if all classes are being predicted correctly. The F1 score in cases of multiple classes is using the average of each class's F1 score, meaning even a low occurrence class receives equal weighting.

3.7.3. Regression Evaluation Metrics

In the case of regression tasks for real valued predictions, the following metrics are used. The metrics are selected to be as interpretable as possible, whilst also being widely used in literature and relevant for transport planning.

Mean Absolute Error

The mean absolute error (MAE) shows the average absolute error over the entire dataset (Fürnkranz et al. 2011):

$$MAE = \frac{1}{|dataset|} \sum_{(x,y) \in dataset} |y - f(x)|\,. \tag{3.13}$$

MAE has thus a range between 0 and positive infinity, with 0 being a perfect prediction on the dataset. The advantage of MAE is that it shows the error in the same units as the targets, veh/h in the case of this thesis, meaning they can be easily interpreted. On the other hand, large errors are not highlighted, even though they are often more relevant.

As the data often contains 0 values, a modified metric is introduced, based on MAE, where only the values with targets with at least 10 veh/h are taken into consideration:

$$MAE_{\geq 10} = \frac{1}{|dataset|} \sum_{(x,y)\in dataset} \mathbb{1}_{\{y\geq 10\}} \cdot |y - f(x)| \,. \tag{3.14}$$

$\mathbb{1}_{\{y\geq 10\}}$ is the indicator function that is 1 when $y \geq 10$ and else 0. This modification focuses on making sure that the model can predict interesting targets, specifically those that have at least 10 veh/h.

R^2

R^2 focuses on how much of the variance in the data is explained by the model. The definition is thus as follows:

$$R^2 = 1 - \frac{\sum_{(x,y)\in dataset}(y - f(x))^2}{\sum_{(\cdot,y)\in dataset}\left(y - \left(\frac{1}{|dataset|}\sum_{(\cdot,y')\in dataset} y'\right)\right)^2} . \tag{3.15}$$

A perfect prediction results in an R^2 of 1, whilst the worst prediction is negative infinity. A value less than 0 means that the model fits worse than a straight line. An advantage of R^2 is that it is easily comparable between models, however, it does require extra care when interpreting linear models, as negative values may indicate an issue with the data.

With the same line of thought as $MAE_{\geq 10}$, a modified R^2 version is implemented, using only target values with at least 10 veh/h:

$$R^2_{\geq 10} = 1 - \frac{\sum_{(x,y)\in dataset} \mathbb{1}_{\{y\geq 10\}} \cdot (y - f(x))^2}{\sum_{(\cdot,y)\in dataset} \mathbb{1}_{\{y\geq 10\}} \cdot \left(y - \left(\frac{1}{|dataset|}\sum_{(\cdot,y')\in dataset} y'\right)\right)^2} . \tag{3.16}$$

An overview of all metrics and losses can be found in Table 3.

In this chapter, an overview of machine learning models that can be used as surrogates is presented. Additionally, regularization techniques and metrics are described. In the following chapter, a formal description of the problem setting is explained.

Table 3 Comparison of the used metrics and losses

Metric/ Loss	Range	Optimal Value	Interpretation
MSE	$[0, \inf)$	0	Average squared error
CE	$[0, \inf)$	0	Average number of bits to identify event
Accuracy	$[0, 1]$	1	Ratio of correct vs total samples
F1	$[0, 1]$	1	Combined precision and recall over all classes
MAE	$[0, \inf)$	0	Mean absolute error in veh/h
$MAE_{\geq 10}$	$[0, \inf)$	0	Mean absolute error in veh/h, only for samples with targets ≥ 10 veh/h
R^2	$(-\inf, 1]$	1	Proportion of explained variance
$R^2_{\geq 10}$	$(-\inf, 1]$	1	Proportion of explained variance, only for samples with targets ≥ 10 veh/h

4. Problem Formalization & Surrogate Model Framework

The following chapter provides a framework of the surrogate idea pursued in this thesis, followed by a formal definition. The formal definition follows a top down approach, starting off with the general definition of transport planning, moving on to the calibrated Four Step Model before diving into the surrogate model approach. Finally, the thesis assumptions and methodological scope is presented.

4.1. Surrogate for the Four Step Model

As described in Chapter 2.4, surrogate modelling has been used to alleviate some of the problems facing complex simulation functions. The author proposes to build a deep learning surrogate model for the Four Step Model, using the recent breakthroughs of graph neural networks to leverage direct graph predictions.

The aim of the surrogate is to improve the downsides of the Four Step Model. Starting with the first shortcoming of assuming the travelers decision making is performed in discrete steps, the graph neural network surrogate solves this by modelling all steps jointly and simultaneously. Additionally, graph neural networks are universal approximators, meaning that they can model any function within a given error (Hornik, Stinchcombe, and White 1989). This implies that with graph neural networks any traveler decision making pattern can be modelled, as long as it is not completely random.

Looking at the problem of deterministic assumption, it is possible to produce confidence scores with graph neural network surrogates. The first approach would be through uncertainty quantification, whilst the second would be by formulating the problem as a classification task. In classification tasks, the target is to predict the traffic buckets, with the neural network providing output probabilities over the buckets. An example for a bucket is $[0veh/h, 100veh/h]$, and for classification a number of buckets need to be defined explicitly.

Due to the nature of neural networks, they implicitly do not require manual calibration, as their parameters are automatically learned from the data. Some hyperparameter tuning and network selection may be needed, however there exist automated algorithms for this as well (X. He, K. Zhao, and Chu 2021). The so called online tuning allows the surrogate to continuously improve itself with automatic data gathering, thus having a model which is always updated to the latest information.

Finally, graph neural networks with GPU acceleration have much lower computational time than traditional simulation techniques, with surrogates experiencing a decrease in computation time on the order of one to two magnitudes (Pfaff et al. 2020).

The surrogate model would be used in practise in the following four steps:

1. First, a massive dataset consisting of millions of samples is generated using existing Four Step Models on cities, to determine the traffic flow given the socio-economic and network data. Note, this step would have to be done only once.
2. Secondly, a graph neural network is trained on the previous dataset. Again, this step would have to be done only once.
3. Thirdly, a traffic engineer takes the pre-trained model, and applies it to a region of interest to directly predict traffic for certain scenarios. The model takes socio-economic and network data as input, whilst producing traffic flow predictions for the entire region as output.
4. Finally, as data is automatically gathered in a specific region, the model is automatically calibrated to fit the new data, thus always being up to date with the current local transport patterns.

A visualization of the steps is shown in Figure 8.

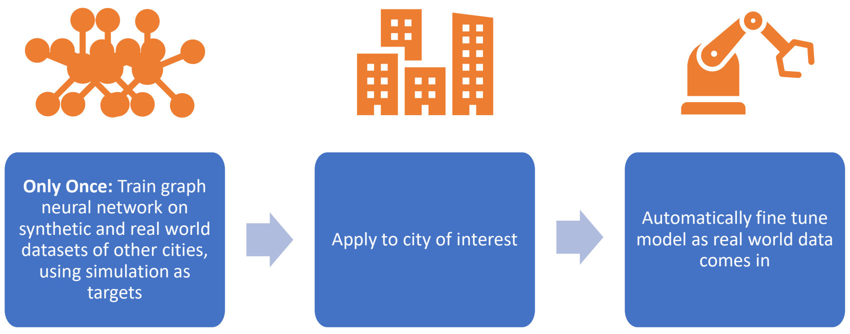

Only Once: Train graph neural network on synthetic and real world datasets of other cities, using simulation as targets
Apply to city of interest
Automatically fine tune model as real world data comes in

Figure 8 How a deep learning surrogate model would be used in practise.

To summarize, the aim is to develop a machine learning surrogate model, which for the same input produces the same output as the original Four Step Model, but with improvements for the problems previously mentioned in Chapter 2.1.2.

4.2. Formal definition

Starting off, the general problem of transport planning in the context of this project is to build a model which takes socio-economic data and network information as input whilst producing traffic predictions that are as close as possible to observed data. The problem can be formalized as following:

$$\begin{aligned} f_{TP} &= \underset{f'_{TP}}{\arg\min}\ e_{\ (\mathcal{S},\ G(\mathcal{N},\mathcal{A}),\ \mathcal{T}) \in D}\ \left(f'_{TP}\left(\mathcal{S},\ G(\mathcal{N},\mathcal{A})\right),\ \mathcal{T} \right), \\ f_{TP} &: \mathcal{S}' \times \mathcal{N}' \rightarrow \mathcal{T}', \end{aligned} \tag{4.1}$$

with $\mathcal{S}'$ being the set of socio-economic data for a given region, $\mathcal{N}'$ being the set of networks with their respective properties, and $\mathcal{T}'$ being the traffic flow along with the network for each mode. $\mathcal{D}$ is the set of real-world, observed data which contains triplets of the socio-economic data ($\mathcal{S}$), network information ($G(\mathcal{N},\mathcal{A})$) and resulting traffic flow ($\mathcal{T}$). The observed data is taken from, for example, traffic flow counts. It is also important to note that the data does not need to be only from a single region of interest but could also contain data from multiple regions or from multiple different observation times. f_{TP} is the optimal transport planning model defined as a function, which is the closest model to the observed data, whilst f'_{TP} is a possibly non-optimal model being evaluated on the observed data. e is an error function over the given dataset, which compares the prediction of the model with the true value. The exact form is defined based on the task at hand, but a standard error function is, for example, the mean squared error (Das, J. Jiang, and Rao 2004).

The exact formalization of the datasets depends highly on both the observed datasets as well as the modelling approach used. In practice, the model that is selected should possess some inherent properties, such as being the most simple yet accurate model within the search space. At this stage, it is also important to note the similarity between Eq. 4.1 and the optimization criteria used in machine learning (Friedman 2017), highlighting that the task comes naturally to machine learning models. For the following chapters, we define f_{4SM} as the best fitting, calibrated to the observed data Four Step Model, defined within the specific context.

4.3. Surrogate Approach & Scope of the Thesis

In the following section, the formal definition for the surrogate model is presented, followed by a general algorithm for how to use it and finally, the methodological scope of the thesis.

4.3.1. Surrogate Model Definition

The aim of surrogate modelling is to find a surrogate $g \in G$ that is as close as possible to the original Four Step Model f_{4SM}. G is the set of functions taking s, n as input and producing t as output, that are cheaper to evaluate than the Four Step Model, can automatically calibrate to data, allow non-deterministic predictions and do joint input modelling. As shown in the previous chapter, graph neural networks possess these qualities. Thus, the search for the surrogate becomes $g \in G_{\textit{GNN}}$. Formally,

$$g = \underset{g' \in \mathcal{G}_{\textit{GNN}}}{\arg\min}\, e_{\,(\mathcal{S},\; G(\mathcal{N},\mathcal{A}),\cdot) \in \mathcal{D}} \;\; \left(f_{4SM}(\mathcal{S},\; G(\mathcal{N},\mathcal{A})),\; g'(\mathcal{S},\; G(\mathcal{N},\mathcal{A}))\right), \tag{4.2}$$

with $e_{(\mathcal{S},G(\mathcal{N},\mathcal{A}),\mathcal{T}) \in \mathcal{D}}$ again being the error over the dataset, but this time comparing the outputs of the Four Step Model with that of the surrogate. The exact error function is defined further in Chapter 3.7, as it is highly dependent on data pre-processing.

Examining Eq. 4.2, it can be seen that the surrogate model is compared to observed data to assess its performance, but this cannot always be done, especially when there is not enough observed data as in the case of the Four Step Model. Thus, to overcome this problem, synthetic or augmented data is employed, denoted by $\mathcal{D}_{\textit{synthetic}}$. Purely synthetic data is used when the entire input data is generated procedurally, whilst augmented implies that a true data source is employed, but modified to produce more data. In this thesis, both types of data generation procedures are used, as shown in the data generation chapter.

Finally, note that any machine learning method can be used as the surrogate model. In this case, in Eq. 4.2, the $\mathcal{G}_{\textit{GNN}}$ class of methods is replaced by the respective machine learning class. The rest of the equation does not change, and the procedure stays identical. This property is used in the next chapter for the baseline models.

4.3.2. Abstract Surrogate Model Algorithm

The abstract algorithm to produce a surrogate, together with fine-tuning on observed data, is defined in Algorithm 1. Note, this is the entire abstract algorithm including data generation, training on the generated data and fine tuning on the observed data, in case it exists.

Algorithm 1 Abstract Surrogate Model Generation Algorithm with Fine Tuning

```
Require: $f_{4SM}$, $\mathbb{G}(f_{4SM})$, $\mathbb{G}_m$, $\mathcal{D}$
  $\mathcal{D}_{\textit{synthetic}} \leftarrow \emptyset$
  while $|\mathcal{D}_{\textit{synthetic}}| \leq \mathbb{G}_m$ do          ▷ 1. Generate synthetic/augmented dataset
      $D_{\textit{synthetic}} \leftarrow D_{\textit{synthetic}} \cup \{\mathbb{G}(f_{4SM})\}$
  end while
  $g \leftarrow t(\emptyset, \mathcal{G}_{\textit{GNN}}, \mathcal{D}_{\textit{synthetic}})$          ▷ 2. Train GNN on synthetic data
  $g \leftarrow t(g, \mathcal{G}_{\textit{GNN}}, \mathcal{D})$          ▷ 3. Fine tune $g$ on observed data, if the dataset exists
  return $g$
```

The algorithm takes as input a tuned Four Step Model (f_{4SM}), a generator ($\mathbb{G}(f_{4SM})$), the number of synthetic data to create ($\mathbb{G}_m$) and a number of true observations (D). The algorithm's output is a surrogate function g, which was initially trained to mimic the provided Four Step Model, and finally is fine-tuned on true, observed data. If there is no true data provided, the algorithm returns a surrogate for the Four Step Model without fine tuning.

Regarding the Four Step Model, it is assumed that it is already fine-tuned and provided as input. This could be interpreted as how expert knowledge of transport modellers is inserted into the deep learning model, for example the travel behaviour difference between North American and European drivers. However, it is essential to keep in mind that if the Four Step Model is too different from true data, the resulting model will also be potentially unstable and produce poor predictions.

Looking at how the algorithm works, the first step is to generate $\mathbb{G}_m$ amount of synthetic samples by calling the $G(f_{4SM})$ function and saving the output to the $D_{\textit{synthetic}}$ dataset. With every call, the generator creates random socio-economic and network data (e.g. an entire city), returning the traffic output of the Four Step Model. After the synthetic dataset is created, the next step is to train a graph neural network on the synthetic dataset, by calling the $t(\emptyset, G_{\textit{GNN}}, D_{\textit{synthetic}})$ function. The empty set denotes that there are no initial models, $G_{\textit{GNN}}$ shows that that the surrogate should be from the class of graph neural networks, and the $D_{\textit{synthetic}}$ is the training data. Finally, after training on synthetic data, the model is fine-tuned on true data with the final t training function call, using the pre-trained model g as the initial configuration.

The assumptions and implementations of the G and t functions will be discussed in the data generation and surrogate models chapters respectively. Note, that as described in the previous section, any machine learning model can be used as a surrogate, but the focus of this thesis is on GNNs.

4.3.3. Assumptions & Methodological Scope

Due to time constraints, the scope of the method has to be limited in this thesis. For this, the following aspects are assumed:

- Travelers' behaviour can be modelled by a function.
- As the experiments in this thesis are limited in resources and there is no observed data available, networks are generated between 15 and 80 nodes, and it is assumed that they are able to represent small networks realistically, whilst the results on these networks can be generalized to larger networks.
- When extracting random subnetworks from a large network, the subnetworks are distinct and realistic when used with the Four Step Model.
- For the data generation, random procedural methods are able to represent real networks, both for network and socio-economic data generation.
- The data generation coefficients produce realistic proxies for cities.
- Due to the absence of a dataset of observed transport values, fine tuning on observed data is not performed but the surrogate model is still valid.

- A single, fixed and minimal Four Step Model is used, which does change between input networks. The Four Step Model is powerful enough to show all aspects of the four stages.
- The Four Step Model is calibrated and represents reality.
- To be useful in practise, an average relative error of less than 10% is needed for the surrogate model with respect to the original Four Step Model, with a focus on high usage links. The value is chosen as it is assumed at this error level the surrogate could be trusted by modelers, at least in basic scenarios.

The assumptions bring the scope of the surrogate in this thesis to a proof of concept. It is attempted to keep the assumptions to a minimum, so as to be able to generalize the results. The key assumption posed here is on the data generation scheme, but it is necessary due to the lack of data, and in future surrogate models, it can be replaced without reducing the validity of the general approach. Based on the assumptions, the approach that is implemented in this thesis is shown in Algorithm 2.

Algorithm 2 Surrogate Model Generation Algorithm - Proof of Concept

```
Require: $f_{4SM}, \mathbb{G}(f_{4SM}), \mathbb{G}_m$
  $\mathcal{D}_{synthetic} \leftarrow \emptyset$
  while $|\mathcal{D}_{synthetic}| \leq \mathbb{G}_m$ do          ▷ 1. Generate synthetic/augmented dataset
      $D_{synthetic} \leftarrow D_{synthetic} \cup \{\mathbb{G}(f_{4SM})\}$
  end while
  $g \leftarrow t(\emptyset, \mathcal{G}_{GNN}, \mathcal{D}_{synthetic})$          ▷ 2. Train GNN on synthetic/augmented dataset
  return $g$
```

The algorithm shown here is identical to Algorithm 1, however, without performing fine tuning. By applying the more basic algorithm, a proof of concept can be developed within the required time frame.

To summarize, the chapter presents an approach to solving the Four Step Model's downsides using a surrogate model, together with the main assumptions for the proof of concept. In the following chapters the exact system is developed.

5. Data Generation & Transformation

A key component of the surrogate modelling paradigm is the generation of a large amount of data. The following chapter presents a description of how the network input data is created, the Four Step Model used for the surrogate training, as well as the steps of the transformation pipeline.

5.1. Data Generation

The following section focuses on the data generation process, which is needed as not enough publicly accessible, standardized data exists to build a surrogate for the Four Step Model. For a surrogate model, it was estimated using existing datasets as references (Dwivedi et al. 2020; Errica et al. 2020) that 10000 samples are needed to create the initial Four Step Model surrogate. Each sample can be considered as a proxy to a single city, containing all network and socio-economic data, together with the resulting transport usage.

5.1.1. Augmented Region Generation

To create realistic, small networks, sub networks are extracted from a real-world network and then apply synthetic zonal data. The following procedure was inspired by previous work in procedural generation, random walks and pathfinding methods (Moore 1959; Parish and Müller 2001; Kelvin and Anand 2020; Bai, Ren, and J. Zhang 2021).

The algorithm works in the following steps, with further details presented below, assuming the number of nodes ($\mathbb{G}_{nodes}$) and network data is provided as input:

1. Load network data.
2. Select a random start node.
3. Perform Breadth-First Search from the start node until $\mathbb{G}_{nodes}$ are reached. Extract all nodes and links within the search space.
4. Select $\mathbb{G}_{zones} = \mathbb{G}_{nodes} * \mathbb{G}_{zones/nodes}$ number of zones to be created. Find $\mathbb{G}_{zones}$ number of nodes that are the furthest away from each other to become zonal nodes.
5. Generate zonal information (e.g. number of residents, jobs) on the previously selected nodes.
6. Generate $\mathbb{G}_{zones} * \mathbb{G}_{buslines/zones}$ amount of bus lines.
7. Run Four Step Model on the data.
8. Save data and output of the Four Step Model.

$\mathbb{G}_{zones/nodes}$ is set to 0.1, and $\mathbb{G}_{buslines/zones}$ set to 1.0, with the values being found to produce realistic looking regions and assumed to be correct for further processing. The zonal information generation is done by first selecting which zones will be residential focused and which will be focused on having workplaces, though mixed zones are also generated implicitly. Then the total number of workplaces is sampled from a Gaussian distribution, with a mean of 1000 workplaces per zone and a standard deviation of 100. From this, the same number of residents are selected as workplaces. Then, using another Gaussian distribution, the car ownership is determined for each residential zone, with a mean of 25% having no car, and a standard deviation of 10% (Kunst 2021). Finally, shopping is distributed randomly uniformly over the zones.

Regarding the bus line generation process, first two zones are randomly selected based on a distribution proportionate to their normalized priority, to imitate the non-uniform distribution of public transportation often found in real cities (Hong et al. 2019). The priority of a zone is determined by the number of residents and workplaces. Then, the shortest route is calculated between the two zones, with bus stops added at both endpoints. The process is finally repeated $zones$ times. By using $zones$ amount of bus lines, on average each zone has one bus line, ensuring that mode choice is a major component of the data.

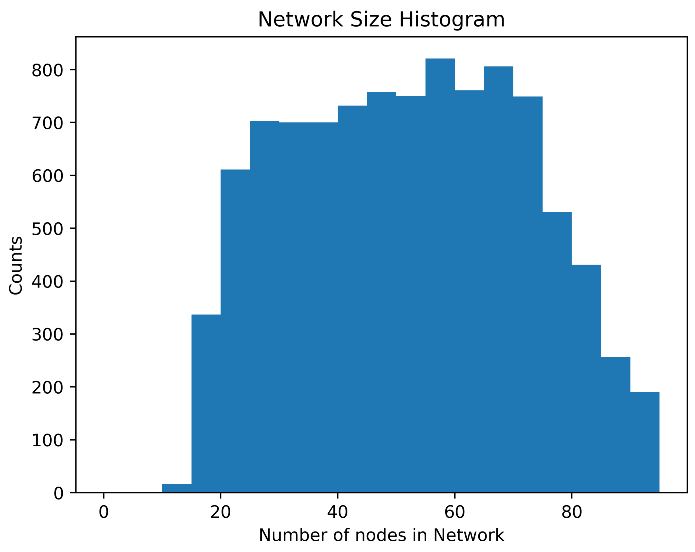


Figure 9 Augmented dataset histogram of graph sizes. Note, the network size is not distributed uniformly as the algorithm prioritizes realistic networks over the exact number of target nodes.

The source network data is provided by the Travel Behavior Institute of the Technical University of Munich, which is initially gathered over the state of Bavaria from Open Street Map (OpenStreetMap contributors 2017). Further processing is then applied, and finally, the network is filtered only to include the major links. The resulting network has 68424 nodes and 138080 links. Further simplifications are conducted by having only two road types, each with one lane: low capacity and high capacity. Low capacity roads have a capacity of 500 veh/h at a maximum speed of 30 km/h, whilst high capacity roads allow 2000 veh/h at a maximum speed of 50km/h. The simplification is conducted to ease the problem for the neural network surrogate whilst developing the proof of concept, and future work can explore a higher road type variety. If no further references are provided, the coefficients, such as $\mathbb{G}_{zones/nodes}$, are determined by experimenting and finding a configuration that produces realistic results based on experts examining a random subsample.

The generation procedure is implemented in Python 3.8 (Van Rossum and Drake 2009), using an API to communicate with PTV Visum 2021 Student Edition (PTV Group 2021). The previous algorithm is run 10000 times to create an appropriate amount of training data. Each generation instance is provided with a target amount of nodes to create, with the total number being uniformly distributed between 15 and 80 nodes over all instances. To maximize generation speed, multiple instances of PTV Visum are launched simultaneously. The total running time for one dataset of 10000 samples takes around 26 hours to create on a consumer level Intel i7 with 16 GB of RAM. Finally, each sample's data, consisting of a single city, is saved as a `JSON` file, to be both easily human and machine-readable.

The augmented Munich region dataset, also referred to as the augmented dataset, is generated with a target number of nodes between 15 and 80. 2 samples have to be removed due to errors, e.g. due to PTV Visum crashing, resulting in a usable dataset of 9998 samples. A histogram of the network sizes is provided in Figure 9. Note, this is not a perfectly uniform distribution, as the algorithm to extract networks prioritizes more realistic networks rather than extracting the exact number of nodes. The prioritization is based on using breadth-first search, adding depth until the number of nodes is within the given range. Stopping the algorithm in between depth stages could be possible, but could result in networks with more dead ends. Finally, the dataset is split into 6398 samples for the training, 1600 for the validation and 2000 for the test dataset, motivated by the splitting values often used in practise in machine learning (Reitermanova 2010). The dataset splits are kept constant across all experiments. An example of a sample of the augmented dataset is shown in Figure 10.

5.1.2. Synthetic Region Generation

In the following paragraphs, an *alternative* data generation algorithm is presented, to generate more training data without using any original network as a source.

It is often seen that more data significantly helps deep learning models (J. Wang, Perez, et al. 2017). On the other hand, the original network data source for the augmented dataset has only a finite number of possible subnetworks, until the training dataset has a critical overlap with the validation and test datasets. To overcome this issue whilst still providing arbitrarily more data, a synthetic region generation algorithm is proposed and implemented, inspired by (Parish and Müller 2001; Smelik et al. 2014; Cogo et al. 2019).

The algorithm performs the following steps, assuming the target amount of nodes ($\mathbb{G}_{nodes}$) is provided as input, with the components for generating zones, zonal information and bus lines

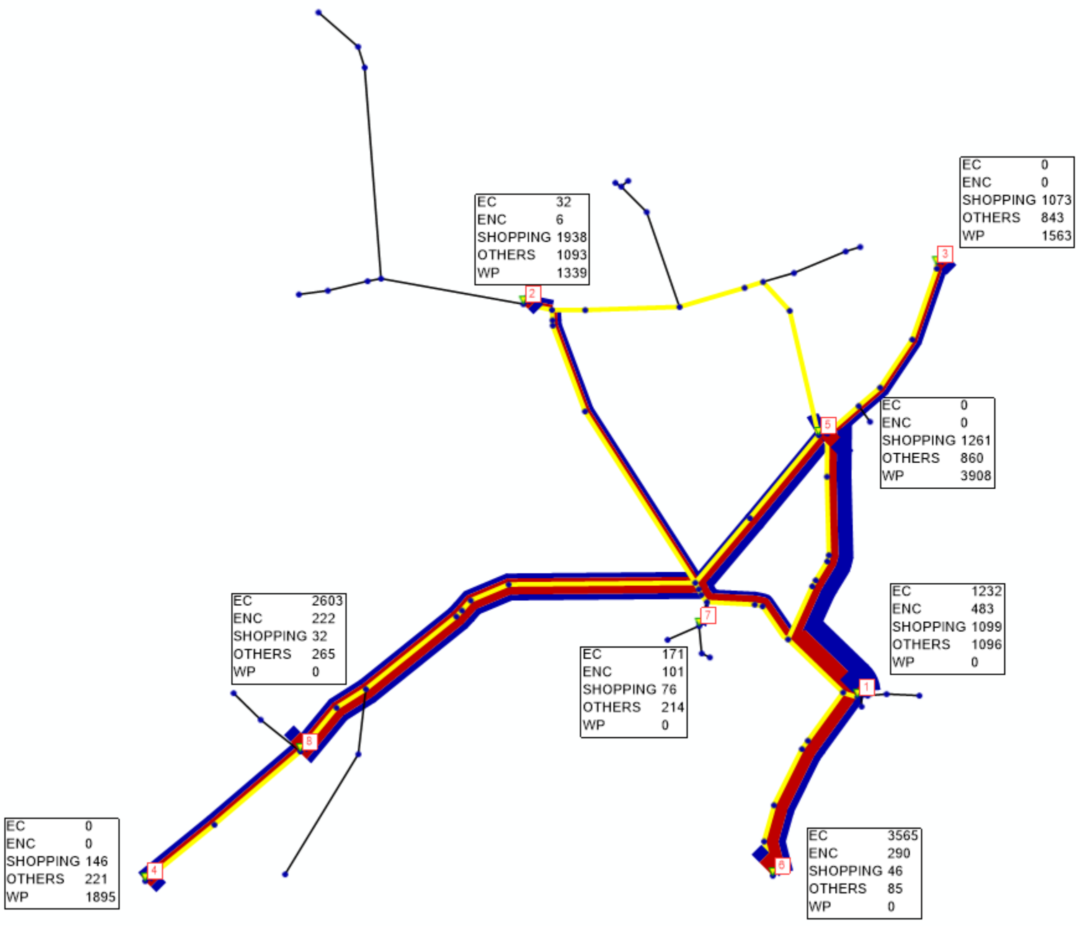


Figure 10 Example of a single sample from the augmented dataset, as seen in PTV Visum 2021. Dots denote network nodes, black lines show links and yellow lines are bus lines. Thicker red blocks around the links show a higher amount of private transport usage, whilst thicker blue links show a higher amount of public transport usage. The black boxes depict zones. The zone parameters are as following: EC denotes number of people employed with a car, ENC is employed without a car, SHOPPING is the amount of shopping places and WP is the amount of workplaces. OTHERS is used in the Halle model to aggregate minor external factors, such as tertiary sector jobs, but is ignored in this model.

identical to that of the augmented generation procedure.

1. Create $\mathbb{G}_{nodes}$ number of random nodes within a square region

2. Select $\mathbb{G}_{zones} = \mathbb{G}_{nodes} * 0.1$ number of zones to be created. Find $\mathbb{G}_{zones}$ number of nodes that are the furthest away from each other to become zonal nodes.

3. Generate zonal information on the previously selected nodes.

4. Generate routes between the zones in two steps.

   a. Create $\mathbb{G}_{zones}$ number of routes. Each route is created by first randomly selecting two zones based on their number of residents and employees. Next, a simple heuristic algorithm starts at a node, examines the closest 7 neighbours, and selects the neighbour which is closest to the target zone as the next step along the route. This is repeated until the route is finished.

   b. Next, if any zone is not yet connected to the network, apply the above routing algorithm to the closest node part of the existing network.

5. Delete any nodes which are not connected.
6. Split long segments of the network until target number of nodes is reached.
7. Generate $\mathbb{G}_{zones}$ amount of bus lines.
8. Save data.

The coefficients used here are also determined based on samples, and are assumed to be realistic for future use. The synthetic dataset is generated also with 10000 samples, with a target number of nodes between 15 and 80. 5 samples are removed due to errors, resulting in a usable dataset that has 9995 samples. A histogram is shown in Figure 11. Note, the synthetic dataset is used in this thesis only as supplementary training data for the augmented dataset with further details being provided in the experiments section.

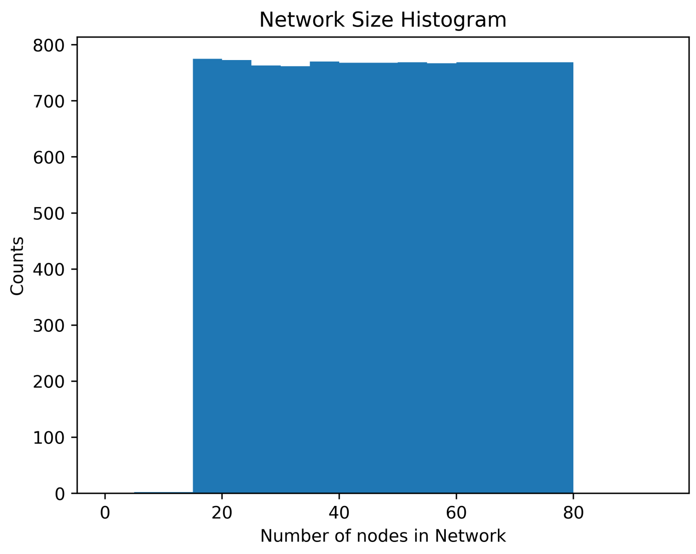


Figure 11 Synthetic dataset histogram of graph sizes. Due to the synthetic node generation procedure, it is possible to generate exact graph sizes.

5.1.3. Four Step Model Implementation

A single Four Step Model generates the resulting transport usage data in PTV Visum 2021 Student Edition (PTV Group 2021). The coefficients and the setup of the Four Step Model is held constant between each evaluation. Future work can explore using different Four Step Models, however, to determine whether a basic surrogate model works, it is essential to keep the original Four Step Model function fixed.

The Four Step Model used in this thesis is based on the Halle example provided by PTV for the zonal parameters used as well as the implementation of the steps, with the exact Four

Step Model coefficients determined by the city of Regensburg, provided by the institute of Transportation System Engineering at the Technical University of Munich. Note, the original Munich network contains only the link network, without socio-economic data. Additionally, the Halle and Regensburg networks are not used due to licensing concerns.

The model is focused on simulating a single AM peak hour, 8am - 9am. There are three modes available: bus, car and walking. The bus mode is denoted as Public Transportation (PuT). Walking can be combined with bus within the simulations for travelers to get to the bus stop, but within this thesis walking is excluded from the PuT volumes used for the surrogate model. The car mode is represented as Private Transportation (PrT). There are four socio-economic variables at each zone: number of households with a car, number of households without a car, amount of work area, amount of shopping area. For the volume delay function, which is used in the Four Step Model to simulate the delay experienced by travelers given a specific travel volume, a standard BPR function is used.

Regarding the four step procedure, first trip generation is performed using standard cross-classification. Next, trip distribution is applied with a gravity model, using the default PTV Visum 2021 options. After that, mode choice is conducted using a logit model. For private transportation, a stochastic traffic assignment step is performed. Next, a loop to the trip distribution step is applied, run until the private transportation assignment converges, or a maximum of 5 iterations have been completed. Finally, outside the loop, the public transport is assigned using headway assignment, assuming a 20-minute headway on all bus lines.

If otherwise stated, all options are used as provided by default by PTV Visum. The exact coefficients can be found in the repository presented in Chapter 5.3 in the `basic-canvas-full4SM.VER` file. All interactions with PTV Visum are done via the Python interface. Future references to a Four Step Model, unless specified otherwise, refer to this implementation.

To summarize, two different data generation schemes are presented, together with a basic Four Step Model that is going to be the target function for the surrogate model. The regions that are modelled assume simple rules, but are complex enough to require all major modelling capabilities of the Four Step Model, resulting in a dataset geared towards developing a proof of concept surrogate.

5.2. Data Transformation

Having the raw data generated, it is essential to be able to use it in deep learning models. In the following section, key data transformation steps are highlighted, proposed and examined. Note, that data transformation and data processing are used synonymously.

Deep neural networks are incredibly sensitive to input data (Sola and Sevilla 1997). Graph data processing is especially complex, as most graph neural networks can only process input data from nodes, not from links. On the other hand, graph neural networks are incredibly new, and there is no publicly used dataset similar to the one generated for the surrogate model. Thus, the requirements for the data transformation step are to allow easy experimentation with different processing steps whilst at the same time being easy to debug.

5.2.1. Data Transformation Pipeline

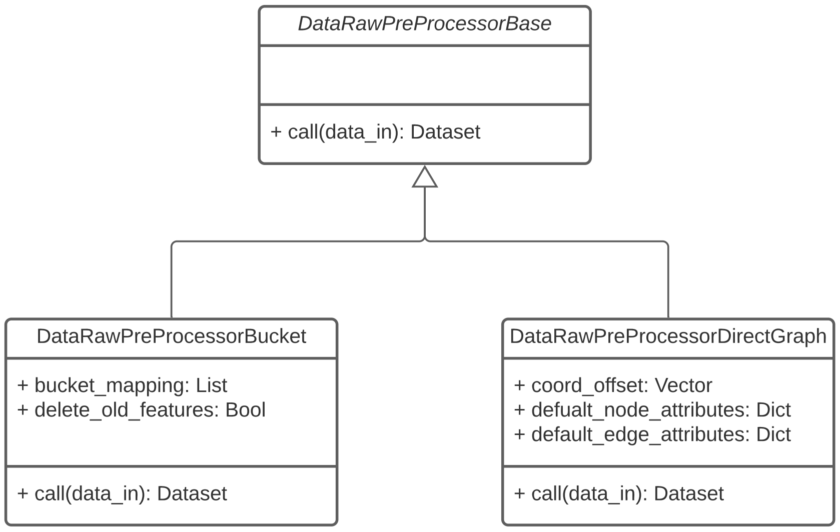
To allow quick experimentation, a modular approach is taken to the data transformation pipeline, described in the following paragraphs. After doing a brief requirement analysis, a system is proposed containing five distinct steps:

1. Data loader,
2. Data graph processing,
3. Numpy conversion,
4. Final processing,
5. Data splitting.

In the first step, the data is loaded from the previously generated `JSON` files, and converted into a list of graphs using the Python library NetworkX (Hagberg, Swart, and S Chult 2008). By using NetworkX, it is possible to easily manipulate and view the individual graphs. An essential aspect for experiment consistency is that the order of the list is kept the same. From now on, the list of graphs is denoted as the dataset.

In the second step, various graph processing methods are applied. A parent class is created for all graph processing steps to facilitate modularity, with a single processing function that takes a dataset as input and produces a modified dataset as output. Then, each method used has its own class, inheriting from the parent class and overwriting the processing function with the logic that is needed. A UML class diagram with two example functions is shown in Figure 12. The result is that multiple steps can be chained together and easily exchanged, for example by first converting the graphs into an appropriate data type, and then bucket the target values into bins.

In the third step, a converter transforms the data from the NetworkX graphs into Numpy arrays. Numpy is a Python library that allows easy and efficient manipulation of arrays (Harris et al. 2020). The Numpy arrays are needed, as all machine learning techniques take only them as input data. Additionally, this step splits input and output data, double-checking that there isn't any overlap. In the fourth step, once all the data is in a Numpy array, final normalization techniques are applied, as neural networks perform best when the data is standardized between -1 and 1.



DataRawPreProcessorBase
+ call(data_in): Dataset
DataRawPreProcessorBucket
+ bucket_mapping: List
+ delete_old_features: Bool
+ call(data_in): Dataset
DataRawPreProcessorDirectGraph
+ coord_offset: Vector
+ defualt_node_attributes: Dict
+ default_edge_attributes: Dict
+ call(data_in): Dataset

Figure 12 UML class diagram of how the data transformation steps are implemented. The `DataRawPreProcessorBase` class is the base abstract class, whilst the example classes `DataRawPreProcessorBucket` and `DataRawPreProcessorDirectGraph` implement the bucketing and graph conversion functions respectively. By using the same function `call` in all steps, it is possible to chain the classes together. Note, there are altogether 29 implemented modules used in the transformation pipeline.

Finally, in the fifth step, the data is split into train, validation and test datasets. All training is done on the train dataset, with model tuning and comparison done on the validation dataset. Only once the model is fixed, and the best approach determined, an evaluation is performed on the test dataset to determine the final score of the model. Using this approach, the least amount of bias is introduced into the model, and the test score can be used to evaluate the model's generalization. Having the modular pipeline enables both fast experimentation as well as consistent results.

5.2.2. Graph Transformations

As explained above, many modules are implemented as part of the data transformation pipeline. A current major limitation of graph neural networks is that most approaches can only use node data as input, and more importantly, only do predictions on nodes. This is a critical problem in the context of this thesis, as the aim is to predict transport usage along the links. Thus, a significant step in the data processing pipeline is to transform the graph so that it is possible to use link information for both input and prediction. Within the scope of this thesis, two approaches are tried out, explained in the next two paragraphs. The transformations are used and compared in various experiments in further chapters.

Line Graph Conversion

The first approach is to use directed line graphs (Harary and Norman 1960). The intuition of the method is as follows, and a formal definition can be found in the original publication. First, for each link in the original graph, a new node is created in the transformed graph. Next, if

Table 4 Comparison of the Line Graph and Direct Graph transformations.

Graph Transformation	Nodes Transformed	Links Transformed	Interpretability	Literature
Line Graph	Original nodes removed; properties propagated to adjacent links	Original links become nodes	Poor	(Harary and Norman 1960; Ramadan et al. 2020; Xiong et al. 2020)
Direct Graph	Original nodes stay; fill irrelevant properties with 0s	Original links also become nodes; fill irrelevant properties with 0s	Easy	(Q. Zhang et al. 2018)

two links previously were connected by a middle node, then their respective nodes of the two links in the transformed graph are also connected. Regarding feature information, each node copies its information to all outgoing edges. This decision is arbitrary, and other approaches to using node information can be used.

Direct Graph Conversion

Due to the complexity of the line graph conversion, a different approach is proposed, inspired by the data transformation in (Q. Zhang et al. 2018), specific for this task. Here, for each link, a new node is generated, with the information being transferred. This requires careful monitoring of which transformed nodes were originally links but is easy to interpret and visualize. Additionally, as nodes do not carry the same information as links, the input data has often many zeros, which could potentially cause convergence problems with some machine learning algorithms. A comparison of the two approaches is shown in Table 4.

Altogether, a total of 29 modules are implemented as part of the data transformation pipeline, and an overview of the transformation topics is presented in Appendix A. To summarize the data chapter, two methods are proposed to generate the input data for the Four Step Model. Then, a basic and fixed Four Step Model determines the transport usage volumes, that are used as targets for the surrogate models. Finally, the data transformation pipeline transforms the raw data into datasets that can be used by deep neural networks.

5.3. Implementation

The implementation of the entire pipeline is performed in Python. All GNNs are implemented in PyTorch Geometric (Fey and Lenssen 2019), which is an extension to the PyTorch library (Paszke et al. 2019). PyTorch contains a number of functions which enable quick development of deep learning models, such as automatic differentiation and prebuilt layers. PyTorch Geometric then extends this by adding in graph specific functions, and especially implementations for many GNNs, including GCNII and GATv2.

The entire project is located as a GitHub repository, found at `https://github.com/nikita68/4_step_model_surrogate`. Here, the entire code can be found, together with its development history. Due to privacy concerns, the data is not provided in the repository, but can be reproduced using the code there, or by contacting the author. A detailed introduction is provided in the readme files in the repository itself, however a brief overview is also provided here.

There are 5 folders containing the most important parts of the project.

- `data_generation` - Contains the data generation code, divided into the synthetic and augmented algorithms. Each version has its own implementation, and subfolders containing the resulting data.
- `data_importer` - Contains all data processing classes, with a small demo showing how to use the generated data in Python in the `basic_demo` subfolder.
- `experiments` - All experiment files and further helpers are located here. Each experiment has a single experiment management file, sorted by data and then by date.
- `literature_review` - The literature review overview is kept track of here, with the main information located in `literature_review.xlsx`
- `organisation` - The organisational information is kept track of here, especially the timeline and presentations.

Further details are provided in the repository, especially as to how to run the individual experiment files and how to locate their respective outputs.

6. Experimental Setup

To answer the primary research question, it is necessary to run experiments and gather results. The primary research question is broken down into tasks, and the tasks are then answered by specific experiments. An overview is provided in Table 5.

Table 5 Overview of all experimental setups, together with their key parameters. BL is abbreviated for baselines, Augmented refers to the augmented dataset, whilst synthetic means the synthetic dataset. A dataset in parentheses means that it is used in a subset of runs.

Setup Chap.	Results Chap.	Aim	Models	Data	Metrics
		6.1 Basic Transport Planning Surrogate			
6.1.1	7.1.1	Find optimal graph transformation strategy	BL, GCNII	Augmented	F1
6.1.2	7.1.2	Predict PrT and PuT jointly	BL, GCN	Augmented	F1
6.1.3	7.1.3	Find optimal models & validate results for basic task	BL, GCNII, GATv3, GCN+GAT	Augmented	F1
		6.2 Regression Predictions			
6.2.1	7.2	Predict exact link values using regression	BL, GCNII	Augmented	$R^2_{\geq 10}$, $MAE_{\geq 10}$
6.2.2	7.2	Predict exact link values using hierarchical regression	BL, GCNII	Augmented	$R^2_{\geq 10}$, $MAE_{\geq 10}$
		6.3 Fine Grained Classification			
6.3.1	7.3.1	Predict exact values using fine grained bins	BL, GCNII	Augmented, (Synthetic)	$R^2_{\geq 10}$, $MAE_{\geq 10}$
6.3.2	7.3.2	Validate bin & additional data strategy	BL, GCNII, GATv3, GCN+GAT	Augmented, Synthetic	$R^2_{\geq 10}$, $MAE_{\geq 10}$

The table includes for each task the respective experiment parameters, focusing on the models, data and metrics, as well as the key aim of the experiments. Both the exact setup and results section are referenced. The tasks and experiments themselves are run sequentially, meaning that they often build on previous findings. In the following chapter, details are provided on the experimental setups, with the results then presented in Chapter 7. All experiments are performed on a consumer-level Intel i5 processor, 16 GB RAM and Nvidia GTX 1660 Super graphics card with 5.9 GB of memory.

6.1. Basic Transport Planning Surrogate

The first task is to find explore whether an extremely basic surrogate model can be built. To answer this, the following experiments focus on performing classification of the traffic level along the links. As the model needs only to predict one of a few buckets, the classification task is easier than directly trying to prediction using regression, thus suited better for evaluating whether a basic model can be built. The bucketing system is also inspired in part by the levels found in congestion warning systems (Gerstenberger, Hösch, and Listl 2018). In this section, only the augmented dataset is used, as it contains more realistic data. The results for the experiments are found in Chapter 7.1.

To use the data in a classification task, the real valued transport data needs to be inserted into a bucket, with the bucket index then being the target. The buckets used for **private transportation (PrT)** are $[0\,veh/h, 10\,veh/h)$, $[10\,veh/h, 500\,veh/h)$, $[500\,veh/h, \infty\,veh/h)$. The three buckets are chosen such that each bucket has over 10% of the samples whilst still being easily interpretable, with a 62%, 21% and 17% distribution respectively on the validation dataset. For **public transportation (PuT)**, the buckets are $[0\,veh/h, 10\,pers/h)$, $[10\,pers/h, 300\,pers/h)$, $[300\,pers/h, \infty\,pers/h)$, with a distribution of 73%, 15% and 12% respectively on the validation dataset. In both cases the buckets correspond to "no transport usage", "low transport usage" and "medium/high transport usage". Regarding the graph diameter, representing the largest amount steps needed to walk along the graph between any two nodes, the maximum in the dataset is 114, whilst the average is 36.

6.1.1. Line Graph vs Direct Graph Conversion

As discussed in the previous chapters, the most advanced GNNs can only take input information and produce predictions on nodes. Thus it is critical to find out how to process the graph data. There are two approaches attempted in this thesis, the line graph and direct graph conversion, introduced in Chapter 5.2.2. The results for this experiment are found in Chapter 7.1.1.

Data

Recall that the line graph approach converts all links to nodes and removes all previous nodes. On the other hand, the direct graph conversion adds links as nodes to the existing set of nodes. In any case, the input for each node is as following, with the differences between the line and direct graph conversion highlighted in bold:

- Whether this is a neighbour of a zone (boolean - **line graph only**),
- Whether this is a zone node (boolean - **direct graph only**),
- X and y coordinates (normalized between 0 and 1),
- Number of employed residents with a car, number of employed residents without a car (standardized),
- Number of workplaces and shopping opportunities, in case of zone nodes (standardized),
- Whether this is a (neighbour of a) stop node for a bus line (boolean),
- Whether there is a bus line going through this link (boolean),
- Whether this is a primary or secondary road (boolean),
- Whether this edge is the forward direction (boolean),
- Length of the link (standardized),
- X and y directions of the link (normalized between 0 and 1),
- PrT capacity and maximum velocity along this link (standardized).

Any properties that are irrelevant, for example number of workplaces for a link, receive the value 0. The target in this case is the PrT usage bucket along the link. Note, there are no PuT predictions done in this experiment, but mode choice still has to be performed implicitly to determine the correct PrT usage. The dataset uses only the augmented dataset. The input variable setup is held constant to all experiments, based on whether a direct or line graph conversion is used.

Models

The first models used here are the three baseline models: majority classifier, random forest and MLP, as defined in Chapter 3.2. The important part to note is that none of these models use network information as input. In initial experiments, it was found that if network information is provided, none of the baseline models converge. The primary metric for all experiments is the averaged F1 score across all classes, presented on the validation set.

A GCNII model is also evaluated on the task. For the line graph conversion, a grid search is performed to determine the best hyperparameters, across the number of hidden layers $\in$ (5, 10, 50), the size of each layer $\in$ (64, 256), the weighted averaging parameters $\alpha \in$ (0.1, 0.4) and $\theta \in$ (0.5, 1.5) parameters. The best performing model has 10 GCNII layers, each containing 256 units, $\alpha = 0.1$, $\theta = 0.5$ and 2 feed forward layers for the final output. The GCNII is then trained for 300 epochs over the entire training dataset. Further GNN models are not initially examined, as the GCNII does not outperform the baselines, and based on existing literature, the GNNs perform similarly.

An identical model for the direct graph conversion is constructed as in the best performing line graph model, i.e. 10 GCNII layers with 256 units. The identical model setup is carried out to ensure that the models can be compared.

All of the results and findings are presented in Chapter 7.1.1, with the results being presented on the validation dataset, as these are preliminary data focused experiments.

6.1.2. Joint Public and Private Transportation Modelling

The previous experiments have focused entirely on modelling private transport, but public transport usage is also a critical component of the transport planning system. In this experiment, it is attempted to build a joint model of the public and private transport. The challenge aims to simultaneously classify the PrT and PuT along each link into their respective three buckets, as defined in the introduction of this section. The results of the experiment are found in Chapter 7.1.2.

The data is processed using direct graph conversion, but with the addition of public transport bins as targets. The result is that each link has two distinct targets, one for PrT and one for PuT. The baseline models are also in the same configuration as the previous experiments, though having two different targets to simultaneously classify. Note, only baselines models and a GCN were used, as the experiments suggest poor convergence for all models. The targets are trained jointly for every approach. Specifically for the GCN, the difference between PrT and PuT outputs are only in the last two feed forward layers. Again, average F1 score across all classes is used as the evaluation metric.

6.1.3. Comparison of GCNII, GATv2, GATv3 and GCN+GATv3

Having seen in the previous experiments that the GCNII model works well in the basic scenario and that PuT and PrT should be predicted using separate models, the following experiment compares whether different GNN models improve the predictions. The results of the experiment are presented in Chapter 7.1.3.

The augmented dataset is used as in the previous experiments. The main difference is that in the final step, based on whether the current training run is aimed at PrT or PuT, the irrelevant target is filtered out. Again, in both cases there are the three buckets, as described in the beginning of this section. All models in this experiment are evaluated and presented on the average F1 score of PrT and PuT on the test dataset.

The same model setup is used for the baseline models as in the previous experiments. For the GCNII a grid search is performed over the number of GCNII layers $\in$ (5, 10, 50, 70), the size of each layer $\in$ (64, 256, 512), $\theta \in (0.5, 1.5)$ and $\alpha \in (0.1, 0.4)$. The upper limits are set due to the resource limitations of the experiment machine. The best performing model on the validation dataset is presented, which is based on 70 layers, each layer containing 512 hidden units, $\alpha = 0.1$ and $\theta = 1.5$. If not stated otherwise, the same parameters are used for all future GCNII experiments.

In addition to using GCNs, the experiment investigates whether more expressive GAT models improve predictions. In preliminary experiments, it was observed that the GATv2 model had convergence problems when attempting to stack more than 5 layers, supported by observations in the literature (M. Chen et al. 2020). Due to the average graph diameter of the dataset being 36, it is mathematically impossible for a shallow GATv2 model to correctly capture the underlying travel patterns. Thus, a GATv3 grid search is performed for the PrT task, with either 20 hidden layers and 512 hidden units or 40 hidden layers with 256 hidden units. These configurations are selected to fit into the memory of the GPU. Based on the validation dataset, the best configuration is 20 hidden layers with 512 hidden units, 2 attention heads averaging their outputs, with the configuration being further used for the PuT model.

Finally, to explore whether the over-squashing and over-smoothing problems described in the previous chapters can be overcome, the combined GCNII and GATv3 model is set up. Here, two configurations are explored for PrT, either 5 or 10 layers are GATv3, using the zone connected graph, whilst the following 60 layers are GCNII layers using the full network graph. The best model contains 10 GATv3 layers and is also used for the PuT setup. Deeper models do not fit into the memory constraints of the experiment machine. The results are seen in Chapter 7.1.3.

6.2. Regression Predictions

For a complete surrogate model, it is essential to find out whether exact values can be predicted in a regression setting. For GNNs, this implies only that the final layer needs to be changed, specifically the activation function. However, regression is incredibly sensitive to the data processing. Note, due to the insufficient performance found in the results, only PrT is explored in this section. The following experiment setups first look into a classical regression regime, followed by the introduction of hierarchical regression. The details of the experiments can be found in Chapter 7.2.

6.2.1. GNN Regression

In the initial experiment, a direct regression setting is chosen. Here, the output of the model as well as the target are single real valued numbers for PrT traffic levels. The loss function in the regression scenario is always mean squared error. The experiment results for the models can be found in Chapter 7.2.

Data

The input data is in the identical configuration as the classification tasks from the previous experiment without the bucketing, only using the augmented dataset with direct graph conversion due to its stability. The same validation set is also used here as in all previous experiments to ensure that the results can be compared. Due to the sensitivity of the model to the target data, a number of configurations are tried out, based on best practises from literature (Benoit 2011; Vashisht 2021). The various modifications are denoted as following:

- log applies the natural logarithm to the targets.
- $masked$ denotes only targets with at least 10 veh/h being used.
- $linear\ n$ uses linear scaling to multiply the targets with $\frac{1}{n}$. If applied together with log, it first applies linear scaling.

Modifications are in some cases combined together, using the "+" symbol to denote this.

Models

Again, the baselines are used to verify improvements by the GNNs on the dataset. The null model in this case is the mean regressor, taking the average of the training dataset as its output. The Random Forest is used in the regression configuration, whilst the MLP is trained using MSE.

Looking at GNNs, the GCNII is exclusively used to improve experimentation speed. The same basic configuration is used that is found optimal in the previous experiments, with 70 GCNII layers, each with 512 hidden units, $\alpha = 0.1$ and $\theta = 1.5$. The models are trained for 300 epochs. Additionally, some modifications are explored, denoted in the results with the following names.

- $GraphNorm$ - Additional GraphNorm layer after every GCNII layer, before the activation function.
- $Res - n$ - With n number of residual feed forward layers, as defined in (K. He et al. 2016), which applies more processing but with leading to GPU requirements.
- $Dropout - n$ - Apply dropout with n percent of units being masked out, though with $n > 25$ in combination with $GraphNorm$, convergence was problematic.

The configurations are also combined, as seen in the results section. Due to the large number of different possible configurations and each training run taking around 8 to 12 hours, a truly exhaustive search was not possible. Instead, a manual search is employed, where a number of initial configurations are iteratively improved to maximize the validation score. This strategy is considered standard practise for deep learning models (Bergstra and Bengio 2012; L. Yang and Shami 2020). Within this experiment, the initial runs attempt to find the best data transformation strategy, and the final runs optimize the GCNII configuration.

In preliminary runs, it was quickly found that the large number of 0 veh/h values skew the dataset and result in poor models, when performing regression directly. Additionally, applying the MAE directly also resulted in metrics showing better performance than actually qualitatively observed, as in the application field of transport planning it is most important to have a model that predicts well on non-zero data. Recall that a model is found in the basic surrogate experiments in Chapter 7.1.3, which has a high accuracy for determining [0, 10) veh/h links. Thus, the two metrics used here are $R^2_{\geq 10}$ and $MAE_{\geq 10}$.

As all models do not reach a low error, they are evaluated only on the validation dataset to limit the bias in future experiments. Note, that if too many model decisions are made based on the results on the test dataset, there is a bias that the model fits on the test dataset implicitly, invalidating the final experimental results of future models (Twomey and Smith 1998). Additionally, as observed in the previous experiments, the difference between validation and test results is expected to be minimal. Again, the experimental results can be found in Chapter 7.2.

6.2.2. Hierarchical Regression

Classification has less data parameters to tune, thus making it often more stable than regression, but does not produce exact values. To combine both of the approaches, the hierarchical regression setup is proposed, which to the best of the author's knowledge, does not exist in literature.

The core idea is to first predict the bucket in which the transport usage should be with the classification approach. In the next step, given the bucket prediction, a value is outputted between -1 and 1 with regression, so as to be able to predict exact values for any bucket by applying normalization. Using the bucket and regression prediction, it is then possible to determine the exact value being predicted. The core idea of the approach is to combine the stability of classification with the precision of regression. The results of the experiment is found in Chapter 7.2.

Data

The input data setup is identical to the previous experiment batch, using the augmented dataset with direct graph conversion for the PrT task. The output data is transformed to have two targets: the bucket class and the scaled value within that bucket. The configuration of the output data is thus dependent on the buckets selected. A number of bucket configurations are explored, using the same manual search strategy as employed in the previous regression experiments. The buckets are denoted as following in the results table:

- Bucket 1 - Buckets [0, 10), [10, 500), >500 veh/h - used due to their success in the first section of experiments.
- Bucket 2 - Buckets [0, 100), [100, 500), >500 veh/h - inspired by the first bucket configuration, but to have more equal sized buckets, without skewing the sample distribution too much.
- Bucket 3 - Buckets [0, 10), [10, 100), [100, 200), [200, 500), [500, 1000), [1000, 2000), >2000 veh/h - to exploit the high number of data points available in the first few buckets.
- Bucket 4 - Buckets going in steps of 100 veh/h from 0 to 4500 - Cover all data points with equidistant buckets.

Models

As the base model, again the GCNII is taken with 70 GCNII layers, each with 512 hidden units, $\alpha = 0.1$ and $\theta = 1.5$. The models are trained for 300 epochs, taking around 8-12 hours per taining run. To implement the hierarchical regression, the classification is performed by two feed forward layers followed by a softmax activation function. The argmax is taken of this output, and then an embedding vector of size 32 is retrieved for this class, inspired by a similar mechanism in natural language processing (Bahar, Makarov, and Ney 2020). The embedding vector is then concatenated to the output of the GCNII layers, and is used by two feed forward layers to produce the output of the link using the tanh activation function, for determine the exact value. A formal definition is provided as following.

$$g(x)_i = 0.5 \cdot \left(1 + g^{regr}\left(\left[\boldsymbol{n}_i^K; N_{embedding}\left(g^{class}(\boldsymbol{n}_i^K)\right)\right]\right)\right) \cdot \tag{6.1}$$

$$\cdot \left(b_{g^{class}(\boldsymbol{n}_i^K),2} - b_{g^{class}(\boldsymbol{n}_i^K),1}\right) + b_{g^{class}(\boldsymbol{n}_i^K),1} \tag{6.2}$$

$$g^{regr}(x) = \sigma_{tanh}\left(N_{ff}\left(\sigma_{relu}\left(N_{ff}\left(x\right)\right)\right)\right) \tag{6.3}$$

$$g^{class}(x) = \arg\max_k \left[\sigma_{softmax}\left(N_{ff}\left(\sigma_{relu}\left(N_{ff}\left(x\right)\right)\right)\right)_k\right] \tag{6.4}$$

$g(x)_i$ is the final output of the model, $g^{regr}(x)_i$ is the output of the regression model for link i, $g^{class}(x)_i$ is the output of the classification model for link i, N_{ff} is a single feed forward layer, $\boldsymbol{n}_i^K$ is the output of the last layer of the GCNs. The $[;]$ is the concatenating operator, $b_{g^{class}(\boldsymbol{n}_i^K),2}$ is the upper bucket value predicted by the classification model whilst $b_{g^{class}(\boldsymbol{n}_i^L),1}$ is the lower value. The $N_{embedding}$ function is a differentiable operator that takes a transport usage bucket ID as input and retrieves a differentiable vector out of memory. The aim of the $N_{embedding}$ is that the neural network automatically learns to represent each bucket with its own custom vector, without having to explicitly define this vector. An illustration is provided in Figure 13.

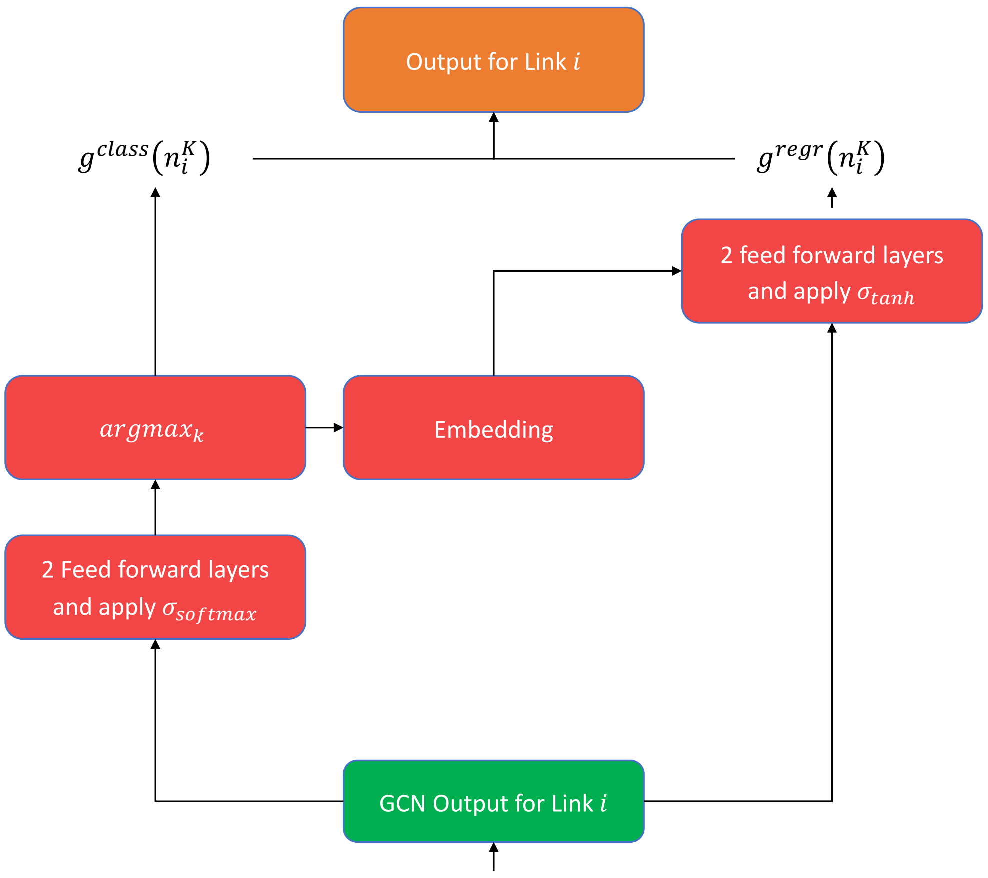

Output for Link i
$g^{class}(n_i^K)$
$g^{regr}(n_i^K)$
2 feed forward layers and apply σ_{tanh}
$argmax_k$
Embedding
2 Feed forward layers and apply $\sigma_{softmax}$
GCN Output for Link i

Figure 13 Illustration of the hierarchical regression model.

Training for the classification is done using cross entropy, whilst the regression component is trained using MSE. The losses are added together to form the final loss, as the losses have similar magnitudes. Different weightings of the loss functions did not show any difference in performance. To stabilize the training, true labels are used as input to the embedding step, but only during the training of the model. GraphNorm was attempted to be used, but resulted in no convergence for the models. Additionally, residual final layers could not be employed as there was not enough GPU memory. Recall that the results are also shown in Chapter 7.2.

6.3. Fine Grained Classification

To exploit the stability of classification, further experiments are conducted to explore fine grained classification as well as a method proposed to convert classification results into real values. The setup is explained in this section and the results are found in Chapter 7.3.

Comparing the F1 score does not make sense when using different buckets, as the the comparison becomes skewed toward the model configurations with fewer buckets and it is not easily interpretable. Thus, it makes more sense to use the same metrics as in the previous section, namely $R^2_{\geq 10}$ and $MAE_{\geq 10}$. However, in classification only probabilities are provided as outputs of the model. To convert the probabilities of buckets into a real value, a small transformation procedure is applied using the expectation operator from statistics. As in the regression section, the focus here is on links with over 10 veh/h in the PTV Visum simulation, due to the fact that a highly accurate model filtering out the lower values is found in the basic surrogate experiments, see Chapter 7.1.3.

First, a uniform distribution across each bucket is assumed and the expectation is taken, resulting the mean value of the bucket. Then, the expectation over all buckets is applied, using the output of the model as the discrete probability distribution. The result is that the model outputs real values in veh/h. Formally,

$$g(x)'_i = \mathbb{E}_{g(x)_i}\left[\mathbb{E}_{uniform}\left[b_k\right]\right] = \sum_k g(x)_i^k \cdot \frac{(b_{k,1} + b_{k,2})}{2}. \tag{6.5}$$

$g(x)'_i$ is the resulting output of the model for the current link i, $\mathbb{E}_{g(x)_i}$ is the expectation using the probabilities provided by the GNN model, $\mathbb{E}_{uniform}$ is the expectation over the bucket k, assuming a uniform distribution. $g(x)_i^k$ is the probability by the GNN model for the current link i for bucket k, $b_{k,1}$ is the lower bound of the bucket whilst $b_{k,2}$ is the upper bound.

In all further experiments, this processed output is used and compared directly to the previous regression experiments, based on the $R^2_{\geq 10}$ and $MAE_{\geq 10}$ metrics.

6.3.1. Optimal Bucket Strategy

First, an optimal bucket strategy needs to be found. A grid search is performed together with a detailed analysis of the best models, with the results presented in Chapter 7.3.1.

The input augmented dataset is identical to the previous experiments, and the target data is always the bucket index for the classification setup. Two bucketing strategies are conducted: equidistant and non-linear bucketing. For equidistant buckets, the following configurations are explored, as they require little reconfiguration:

- *Buckets-e23* - 23 buckets in step size of 200 veh/h.
- *Buckets-e45* - 45 buckets in step size of 100 veh/h.
- *Buckets-e90* - 90 buckets in step size of 50 veh/h.
- *Buckets-e180* - 180 buckets in step size of 25 veh/h.
- *Buckets-e450* - 450 buckets in step size of 10 veh/h.

When looking at the equidistant buckets, it is seen that many of the higher value buckets have no or few training samples, whilst lower values have many thousands of samples. The difference in sample sizes can greatly impact the performance of GNNs, and especially restrict their generalization behaviour. To compensate for this effect, non-linear bucket sizes are also explored, to ensure that all buckets have at least 1 training sample. The following configurations are examined:

- *Buckets-nl38* - Steps of 25 from 0 veh/h to 250 veh/h, steps of 50 from 250 veh/h to 1000 veh/h, steps of 100 from 1000 veh/h to 1500 veh/h, steps of 500 up 4500 veh/h.
- *Buckets-nl54* - Steps of 25 from 0 veh/h to 500 veh/h, steps of 50 from 500 veh/h to 1500 veh/h, steps of 100 from 1500 veh/h to 2000 veh/h, steps of 500 up 4500 veh/h.

The second of the non-linear buckets goes into greater detail, but with less training samples per bucket, and vice versa for the first configuration. Further configurations are not explored as the results imply that there is no significant further improvement.

Synthetic data to increase the training dataset has been shown in existing literature to improve deep learning models (Nikolenko 2021). In parallel to the previous experiments, the synthetic dataset was generated. The synthetic dataset is used only to expand the training data, with the validation and test datasets being kept the same as in the previous experiments to ensure proper comparisons. The new training dataset increases its size to 16393 samples, more than 2.5 times larger than the original dataset. The two previously best performing bucket configurations, nl54 and e90, are then trained from scratch using additional synthetic training data.

The GCNII used in these experiments are based on the previous best performing neural network configuration. Specifically, a GCNII is used with 70 layers, 512 hidden units, $\alpha = 0.1$, $\theta = 1.5$, 3 residual feed forward layers, GraphNorm between after each convolution operation and a dropout of 25%. To improve sensitivity to non-zero values, only training targets with ≥ 10 veh/h are used. As this is a classification setting and the output of the model is based on the softmax, cross entropy is used as the loss function. Each training run of a model takes around 12-15 hours. Additionally, the baselines are also evaluated. Again, the results are described in Chapter 7.3.1.

6.3.2. Validating Non-Linear Buckets & Additional Synthetic Data

To further validate the previous results, the final experiments examine how the GATv3 and GCN+GAT models perform using the training data supplemented by the synthetic samples on both PrT and PuT. The results of the experiments are seen in Chapter 7.3.2.

The data setup is identical to the previous experiment for PrT, but only using the *Buckets-nl54* configuration, as it is the best performing configuration, as seen in Chapter 7.3.1. The buckets go in steps of 25 from 0 veh/h to 500 veh/h, steps of 50 from 500 veh/h to 1500 veh/h, steps of 100 from 1500 veh/h to 2000 veh/h and steps of 500 up 4500 veh/h.

Regarding PuT, an identical strategy to nl54 is taken to ensure enough training samples per bucket, but adjusted to the lower values. The resulting steps are to go in 25 pers/h from 10 to 300, then in steps of 50 pers/h from 300 to 1000, next steps of 100 pers/h from 1000 to 2000 and finally in steps of 500 pers/h from 200 to 4500. Even though the bucket configuration contains 42 buckets, it is denoted as nl54 for simplicity.

The models are based on the best performing regression model which fits into the hardware limitations. The GCNII is used with 70 layers, 512 hidden units, $\alpha = 0.1$, $\theta = 1.5$, 3 residual feed forward layers, GraphNorm after each convolution operation and a dropout of 25%. The GATv3 model contains 20 layers each with 512 hidden units, based on 2 attention heads performing the averaging merging strategy, 3 residual feed forward layer, GraphNorm and dropout with 25%. Finally, the GCN+GAT model is configured with 10 zonal GATv3 layers, each with 128 hidden units, 2 attention heads being averaged together. Following that, 60 GCNII layers are used, each with 512 hidden units, $\alpha = 0.1$, $\theta = 1.5$ and 3 residual feed forward layers. GraphNorm and dropout of 25% is used for each convolution layer. Note, each training run takes around 40 hours, severely limiting the number of possible experiments performed.

To summarize, a large number of experiments are performed, focusing on tackling the surrogate modelling task from different the classification, regression and combined perspectives. Experimental setups are guided by best practises found in literature, whilst also adding novel components in specific situations. In the following chapter, the experiments are evaluated and the results presented.

7. Experimental Results

In the following chapter, the experiment topics are presented, together with their respective setups, results and implications. An overview can be seen in Table 6.

Table 6 Overview of all experimental results, together with their aim and key results.

Results Chap.	Setup Chap.	Aim	Key Results
		7.1 Basic Transport Planning Surrogate	
7.1.1	6.1.1	Find optimal graph transformation strategy	Direct graph conversion significantly better
7.1.2	6.1.2	Attempt to predict PrT and PuT jointly	Does not converge when training jointly
7.1.3	6.1.3	Find optimal models & validate results for basic task	All GNNs substantially better than baselines & model vital transport planning concepts well
		7.2 Regression Predictions	
7.2	6.2.1	Predict exact link values using regression	Better than baselines but predictions not good enough for practical use
7.2	6.2.2	Predict exact link values using hierarchical regression	Worse than regression
		7.3 Fine Grained Classification	
7.3.1	6.3.1	Predict exact link values using fine grained bins	Optimal bin configuration found, better than regression but not accurate enough for practical use
7.3.2	6.3.2	Validate bin & extra data strategy	Extra data helps significantly, but still not at desired accuracy

The experiments were performed sequentially, in the order depicted in this chapter. The experiments are setup as described in Chapter 6, and the respective results are presented here. Note, that the aim of the experiments here is to find the best surrogate proof of concept. Thus, the experiments try out various approaches and not necessarily exhaustively check which configuration is optimal.

7.1. Basic Transport Planning Surrogate Model Results

The first task focuses on building the basic surrogate model, by aiming to classify the traffic level of each link into three buckets. The exact model details are provided in Chapter 6.1. The experiments initially examine the optimal data processing pipeline, before moving on to task configurations and finally a deep dive into the models.

7.1.1. Line Graph vs Direct Graph Conversion Results

As described in Chapter 6.1.1, the first experiment compares the line graph and direct graph conversions and their effect on the outputs. The models are identical, except in that they use different graph processing steps. The results of the training runs of the models are presented in Table 7. It can be seen that for the line graph conversion, the GCNII has marginally worse performance than the baselines. A brief qualitative examination of the line graph results show that no model captures the underlying patterns, and most of the F1 score comes from predicting trivial traffic buckets with no PrT usage. On the other hand, a GCNII with direct graph conversion achieves significantly better results than the baselines, supported by manual examinations of individual outputs.

Table 7 Results on the basic PrT classification task comparing line graph with direct graphs on the validation dataset.

Model	Validation $F1$ - Line Graph	Validation $F1$ - Direct Graph
Majority Classifier	0.25	0.25
Random Forest	**0.54**	0.54
MLP	**0.54**	0.54
GCNII - 10 layers, 256 units	0.53	**0.77**

Interpretation and Findings

Seeing that the direct graph approach is able to perform significantly better than the line graph approach all future experiments should use the method. A hypothesis as to why line graph performs worse, is potentially due to the exponentially increasing number of neighbours that every node receives input from after the transformation. For other GCN types, this has shown to degrade performance (M. Chen et al. 2020).

7.1.2. Joint Public and Private Transportation Modelling Results

Predicting public transportation is critical for a proper surrogate model. In the following experiment, PrT and PuT are attempted to be predicted jointly, with the setup described in Chapter 6.1.2. The result was that none of the models converged, resulting in random outputs.

A hypothesis, in this case, is that PrT and PuT follow significantly different logic on the network, meaning they require a higher amount of processing separation. Thus, it was concluded to separate the PrT and PuT tasks into separate models for future experiments. Note, this does not limit the surrogate model of the Four Step Model, as the models would still implicitly have to perform mode choice.

7.1.3. Comparison of GCNII, GATv2, GATv3 and GCN+GATv3 Results

Having found the optimal data setup, namely with direct graph conversion and splitting up PrT and PuT into different tasks, the following experiment compares various GNN models, to find the optimal configuration for the basic surrogate task. The experiment setup details are found in Chapter 6.1.3.

The results of all models can be found in Table 8. For the GCNII, a grid search is performed over the main hyperparameters. It is observed that the GCNII model performs significantly better than all baselines on the PrT task. Note that the accuracy of the model is at 90%, and most errors are performed in classifying the $[10\,veh/h, 500\,veh/h)$ bin. For the bin of $[0\,veh/h, 10\,veh/h)$, an accuracy of 95% is achieved. It is seen that the configuration containing both the largest number of layers and largest layer size in the GCNII performs best.

It is observed that for PrT the best performing model is the GATv3, followed closely by the GCN+GAT model, and then the GCNII model. For PuT, the best performing model is the GCNII, followed closely by GCN+GAT model and finally GATv3. All three GNNs perform similarly to each other, and significantly better than the baselines. Note, the majority classifier for the PuT is higher than for PrT.

Table 8 Results on test datasets for basic separate PrT and PuT modelling, using more advanced models.

Model	PrT Test $F1$ Score	PuT Test $F1$ Score
Majority Classifier	0.25	0.28
Random Forest	0.54	0.62
MLP	0.53	0.58
GCNII - 70 layers, 512 units	0.85	**0.83**
GATv3	**0.87**	0.78
GCN + GAT	0.86	0.82

The convergence of all GNN models for the direct graph are observed to be stable and level out at roughly 300 epochs. The training loss can be seen for the GCNII in Figure 14, and the validation F1 score in Figure 15. The stable convergence pattern is observed along the other GNN training runs as well. If not otherwise stated, all presented GNN models in this thesis have similar convergence behaviour on the training and validation data.

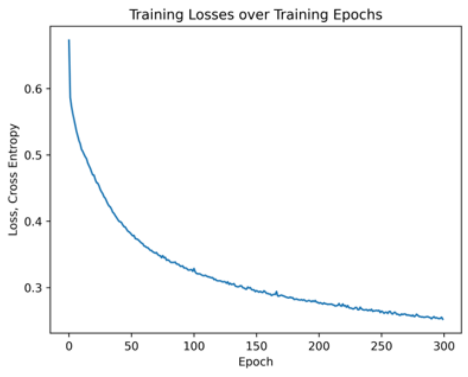


Figure 14 Training loss of cross entropy of the GCNII model over training epochs.

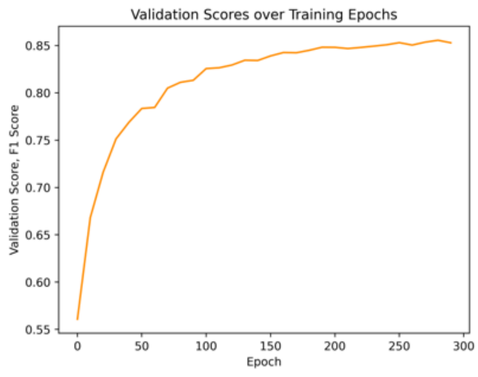


Figure 15 F1 score of the GCNII model over the validation dataset.

Interpretation and Findings

First, due to the higher score of the majority classifier, it can be seen that the PuT task is slightly easier than the PrT classification task. This is also to be expected, as there are more links in the PuT classification containing little to no transport usage.

Based on the improved performance, direct graph conversion shows to be a critical part of the pipeline in building a graph neural network surrogate. All future experiments thus include the direct graph method. It can also be concluded that a larger number of layers and thus more trainable parameters are helpful but are limited by the resources available through the training machine. Additionally, it is observed that all models perform similarly between their test and validation scores, verifying both that an appropriate data splitting mechanism is employed, as well as the generalization of the models is highly probable.

Comparing the various GNNs, all three perform well and highly similar, with only slight variations between the models and classification tasks. First, this implies that deep GNNs are a suitable candidate to further explore for a surrogate model for the Four Step Model. Secondly, it implies that the exact model does not make a major difference. For future experiments, a GCNII can be used for initial experiments, and other GNN models further tested once a potential stable problem setup is found.

Focusing on the GATv3 model, note that it has a depth of only 20 layers, meaning that only nodes and links within 20 steps over the graph of each other can exchange information. The performance does not degrade substantially, which is hypothesized due to the average graph diameter being only 36, meaning that most non-zero nodes are within a shorter distance than this. For larger graphs this could result in problems.

Finally, looking at the GCN+GAT, it has similar performance to the other models. A hypothesis on this is that either it does not solve the over-squashing problem, or the issue does not exist for this initial task. Further research into this is examined in future experiments.

7.1.4. Output Examples

Quantitative analysis shows that the models perform well, however a qualitative examination helps to understand the context of the models. In the following subsection, some examples are shown of the surrogate, highlighting how the basic model covers the main downsides of the Four Step Model, as well as some errors that the surrogate model makes.

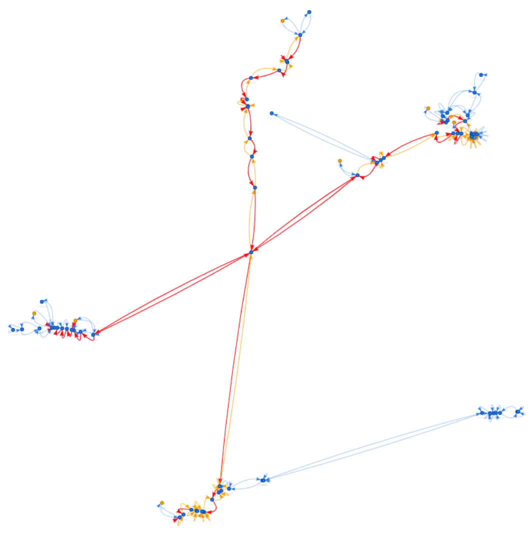


Figure 16 Prediction of GCN+GAT model. The colour of the link is determined as following: blue is [0, 10) veh/h, orange is [10, 500) veh/h and red is over 500 veh/h. Blue nodes are normal nodes in the network, orange nodes are those connected to zones.

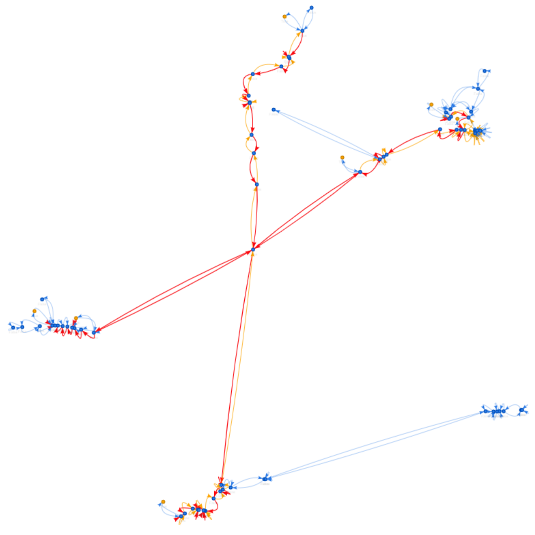


Figure 17 Targets using Visum simulation. The colour of the link is determined as following: blue is [0, 10) veh/h, orange is [10, 500) veh/h and red is over 500 veh/h. Blue nodes are normal nodes in the network, orange nodes are those connected to zones.

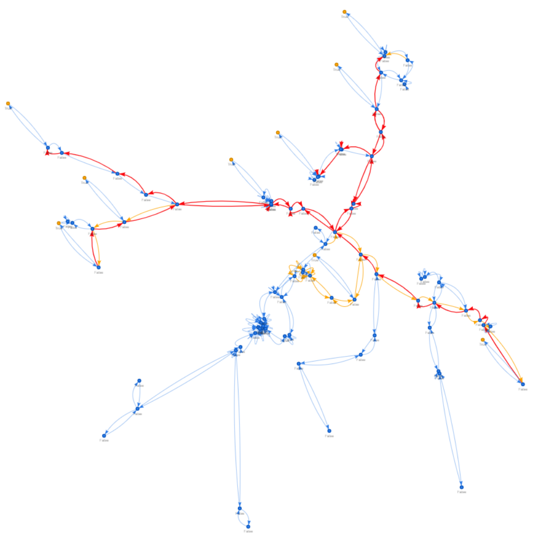


Figure 18 Prediction II of GCN+GAT model. Note the cluster of incorrect predictions near the center of the network. The colour of the link is determined as following: blue is [0, 10) veh/h, orange is [10, 500) veh/h and red is over 500 veh/h. Blue nodes are normal nodes in the network, orange nodes are those connected to zones.

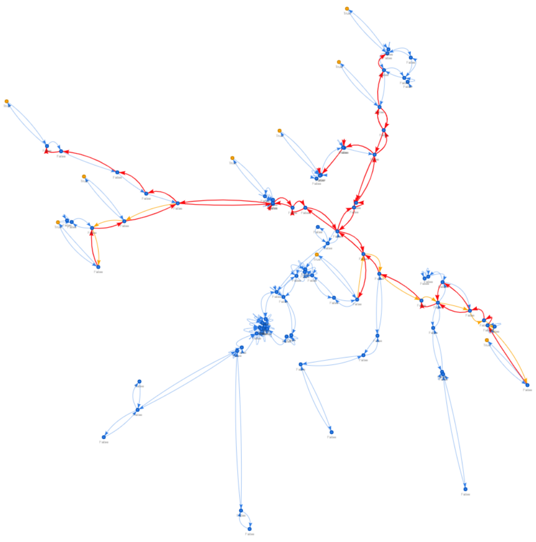


Figure 19 Targets II using Visum simulation. The colour of the link is determined as following: blue is [0, 10) veh/h, orange is [10, 500) veh/h and red is over 500 veh/h. Blue nodes are normal nodes in the network, orange nodes are those connected to zones.

The examples show some randomly selected outputs from the validation dataset of the trained GCN+GAT model from the previous experiment on the PrT data. An example of a prediction is shown in Figure 16, with the respective targets in Figure 17, and another prediction example in Figure 18 with the respective targets in Figure 19.

In general it can be seen that the model understands the major concepts of supply and demand, together with how it is assigned across the network. However, an observation that is found is that errors often propagate across the network. In this setting, an initial link is misclassified and the misclassification then affects a large area of the network. An example is seen in Figures 16 and 17, in the center where a number of "no traffic" links are predicted to have [10, 500) veh/h. This effect is probably due to the way that GNNs make predictions by iteratively propagating the feature vector across nodes.

Congested Network

An important effect in transport planning is that of the congested network, where travelers decide to take an alternative and often longer route to avoid congestion. The trained GCN+GAT model from the previous experiment is able to learn this pattern, and an example is shown in Figures 20 and 21. Observe that in the target of the bottom link going from right to left, there is also a number of travelers taking the same route but longer through the node at the top. The GCN+GAT model is able to distinguish this difference, however, it overpredicts how many travelers take the bottom route.

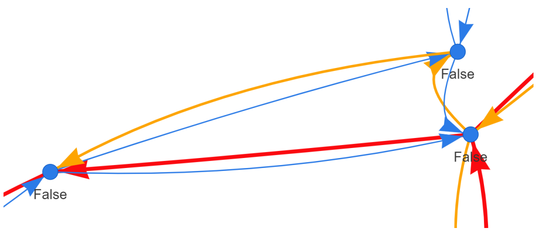


Figure 20 Prediction of GCN+GAT model, able to predict alternate routing. The colour of the link is determined as following: blue is [0, 10) veh/h, orange is [10, 500) veh/h and red is over 500 veh/h. Blue nodes are normal nodes in the network, orange nodes are those connected to zones.

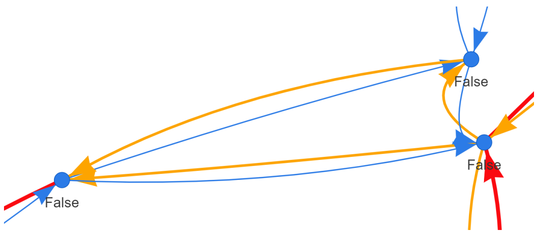


Figure 21 Targets using Visum simulation, with travelers taking alternate routes from the node on the right hand side to the node on the left. The colour of the link is determined as following: blue is [0, 10) veh/h, orange is [10, 500) veh/h and red is over 500 veh/h. Blue nodes are normal nodes in the network, orange nodes are those connected to zones.

Output Probabilities

In the classification problem setting, the output of a GNN model can be interpreted as probabilities, similar to confidence intervals across the classes. An example is shown in Figure 22. Such an ability is important, solving one of the main problems with the Four Step Model by allowing decision makers to understand the model better and thus trusting it more.

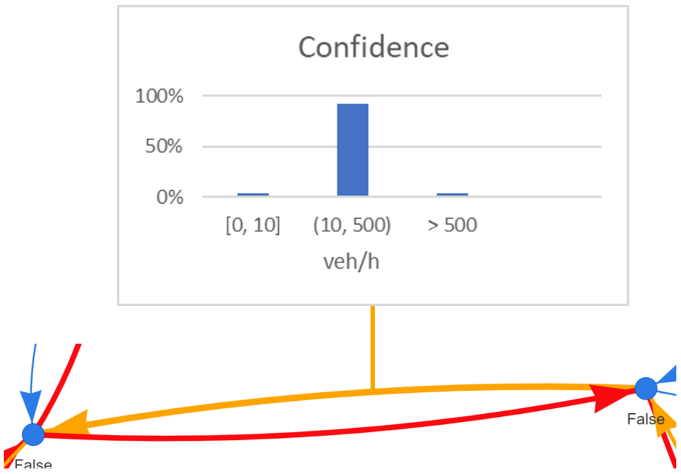


Figure 22 Example of probabilities of prediction for a link.

Computation Speed

Another problem of the Four Step Model is that it has long run times of large models, using CPUs for computation. The runtime for the average Four Step Model in Visum for a large network from the dataset is around 15 seconds, whilst using a CPU the same network takes around 0.3 seconds using a trained GCN+GAT model. The results translates into a 50 time speed up. Further speed up can easily be achieved by using GPUs and batching predictions together.

7.1.5. Basic Surrogate Findings Overview

The answer to whether the most basic transport planning surrogate model can be built is yes. The models perform especially well for the class of [0, 10) veh/h, and provide good predictions for the other two classes, being able to comprehend traffic patterns. It is found that direct graph conversion is important to transform the graph, and deep GNNs perform better than both their shallow counterparts and baselines from machine learning. Additionally, the best convergence is achieved by separating PrT and PuT into individual tasks, though performance between models is highly correlated between the two tasks. Finally, the three tested GNNs perform highly similarly, implying that not all models need to be run for every experiment configuration.

Even though these models are too basic to be used directly for transport planning, they could potentially be exploited to provide an initial assignment for the Four Step Model. The resulting convergence speed could be higher, though more research needs to be conducted. Further experiments now try to expand the models to improve their granularity.

7.2. Regression & Hierarchical Regression Results

Having seeing that the most basic classification setting works, it is important to find out whether exact values can be predicted in a regression setting. The following experiment are based on the setups described in Chapter 6.2.1 and Chapter 6.2.2.

Recall that in the results, log, $masked$, $linear - n$ denote data transformations whilst $GraphNorm$, $Res - n$ and $Dropout - n$ different model configurations, with n being a number provided as a parameter in the results description. The individual configurations are combined with the "+" symbol. Additionally, the four buckets used for hierarchical regression can be summarized respectively as 1) three simple buckets, 2) three relatively regular buckets, 3) seven buckets exploiting the data distribution and 4) 45 equidistant fine grained buckets.

The results of the experiments can be found in Table 9. Note again that manual search is employed to determine the optimal data transformation and model configuration, as is standard practise and described in further detail in the Chapter 6.2.1. Here, the baselines perform poorly, with little improvement by the Random Forest and MLP over the mean regressor. Focusing on classical regression for the GCNII, it can also be seen that applying logarithmic transformations on the targets improves the performance, but not significantly. Masking targets also improves performance slightly. When applying a linear transformation before the logarithm function, the performance improves to around 200 $MAE_{\geq 10}$. However, more significant performance improvement is experienced by only using a linear transformation, reaching 176.6 $MAE_{\geq 10}$. The best performing model additionally uses target masking, *GraphNorm*, a dropout of 25% and 3 residual final layers, to reach a $MAE_{\geq 10}$ of 131.1 and $R^2_{\geq 10}$ of 0.84. It was found that more residual final layers degrade the performance.

Looking at hierarchical regression, all models reach similar performance, but even the best model using "Bucket 3" option reaches only 175.8 $MAE_{\geq 10}$, significantly worse than the best regression model. Comparing bucket sizes, it can be seen that more buckets result in better performance. However, with 45 buckets the $R^2_{\geq 10}$ value does not correlate with the $MAE_{\geq 10}$ value, meaning there may be some discrepancy in the predictions. In summary, even though the hierarchical regression models perform better than the baselines, they do not come close the best model from the classical regression experiments.

To bring the errors into perspective, the average target value in the validation dataset, excluding values under 10 veh/h, is 549.1 veh/h, meaning that 131.1 results in an error of 23.8%. Qualitative sampling of the results, where individual outputs were manually examined, support the inaccuracies reported by the metrics.

Table 9 Results of regression on validation datasets for PrT.

Model	Validation $R^2_{\geq 10}$	Validation $MAE_{\geq 10}$
Mean Regressor	0.00	394.8
Random Forest	0.00	373.1
MLP	0.00	371.6
GCNII + log	0.32	272.6
GCNII + log + $masked$	0.32	267.7
GCNII + $linear - 250$ + $masked$	0.66	176.9
GCNII + $linear - 100$ + log + $masked$	0.59	201.7
GCNII + $linear - 500$ + log + $masked$	0.69	197.1
GCNII + $linear - 250$ + $masked$ + $GraphNorm$	0.80	147.9
GCNII + $linear - 250$ + $masked$ + $GraphNorm$ + $Dropout - 25$	0.81	145.7
GCNII + $\boldsymbol{linear - 250 + masked + GraphNorm + Dropout - 25 + Res - 3}$	**0.84**	**131.1**
GCNII + $linear - 250$ + $masked$ + $GraphNorm$ + $Dropout - 25$ + $Res - 6$	0.83	137.6
GCNII - H-Regr + Bucket 1	0.64	197.5
GCNII - H-Regr + Bucket 2	0.68	180.0
GCNII - H-Regr + Bucket 3	0.81	175.8
GCNII - H-Regr + Bucket 4	0.65	176.7

Interpretation and Findings

The experiments on the regression task have shown that the linear transformation of data is best for strategic traffic prediction. Additionally, target masking, *GraphNorm*, dropout and residual final layers make significant improvements.

However, when looking at the scores of the best model, it is still too inaccurate to be used by transport planners, having an error of 23.8%. On the other hand, in literature there have not been any further proposals as to how to improve performance of GNNs in the classical regression settings. Thus, further experiments go into alternative formulations that could result in a lower average error.

The subpar performance found in hierarchical regression shows that it may be too complex as a problem for the models experimented with here. Deeper models with more parameters could potentially be a solution, but there needs to be further research on this topic. A number of hypotheses are open as to why regression does not perform as well as expected:

- Not enough data - Potentially more training data could help model convergence.
- Models not expressive enough - Deeper models on better hardware could potentially find more subtle patterns.
- MSE not stable enough - In complex training settings like here, MSE might result in poor gradients for the model, also found in literature (Z. Zhang and Sabuncu 2018).

Further research needs to be performed into this area, but the results of the models with a high number of buckets in the hierarchical regression setting seem like a promising aspect to investigate.

7.3. Fine Grained Classification Results

With regression failing to reach good predictions, fine grained classification is attempted to reach better results. In the following experiments, classification is performed on fine grained buckets. By applying the double expectation operator, it is possible to convert probabilities to real values, with all details about the setup found in Chapter 6.3. The following experiments first find the optimal bucket strategy, followed by an investigation into using extra synthetic data.

7.3.1. Optimal Bucket Strategy Results

In the following results, experiments using various bucket configurations are explored, as well as the addition of further synthetic data. The experiments are done only on PrT using the GCNII, with the details of the experimental setup in Chapter 6.3.1.

The results of the experiments can be found in Table 10. Note, that $R^2_{\geq 10}$ is often noisy, making it harder to make exact comparisons. Looking at equidistant buckets, it can be seen that all reach good results, when comparing to the regression experiments, but only GCNII - Buckets-e90 slightly surpasses the $MAE_{\geq 10}$ score of the best regression model. In the non-linear buckets, both configurations reach good results, but do not surpass the best performing model of e90.

The two best performing bucket configurations, nl54 and e90, are then trained from scratch using additional synthetic training data. It can be seen that both models with the extra data perform significantly better than the baselines and have an 16% better performance than the models without the extra data. The models have similar performance, but the Buckets-nl54 model performs slightly better, based on the $MAE_{\geq 10}$ metric.

Table 10 Results of fine grained classification on validation datasets for PrT.

Model	Validation $R^2_{\geq 10}$	Validation $MAE_{\geq 10}$
Mean Regressor	0.00	394.8
Random Forest	0.00	373.1
MLP	0.00	371.6
GCNII - Regression Best	0.84	131.1
GCNII - Buckets-e23	0.79	145.4
GCNII - Buckets-e45	0.78	140.1
GCNII - Buckets-e90	0.81	129.6
GCNII - Buckets-e180	0.78	139.5
GCNII - Buckets-e450	0.78	137.8
GCNII - Buckets-nl38	0.79	136.1
GCNII - Buckets-nl54	0.79	133.0
GCNII - Buckets-e90 - extra data	0.85	114.8
GCNII - Buckets-nl54 - extra data	**0.86**	**111.4**

Looking more closely at the differences between the models "GCNII - Buckets-e90 - extra data" and "GCNII - Buckets-nl54 - extra data", it is found that the GCNII with 90 buckets and extra data predicts high values poorly. As seen in Figure 23, all true values above around 2500 veh/h get assigned values around 2600 veh/h. On the other hand, the equivalent model but using the non-linear 54 buckets continues to predict proportionally high values, as seen in Figure 24.

A visualisation of the number of buckets in comparison to the Validation $MAE_{\geq 10}$ is found in Figure 25, showing that there is an optimal number of buckets. Additionally, an alternative training loss formulation was examined using the expected value and then applying the MSE

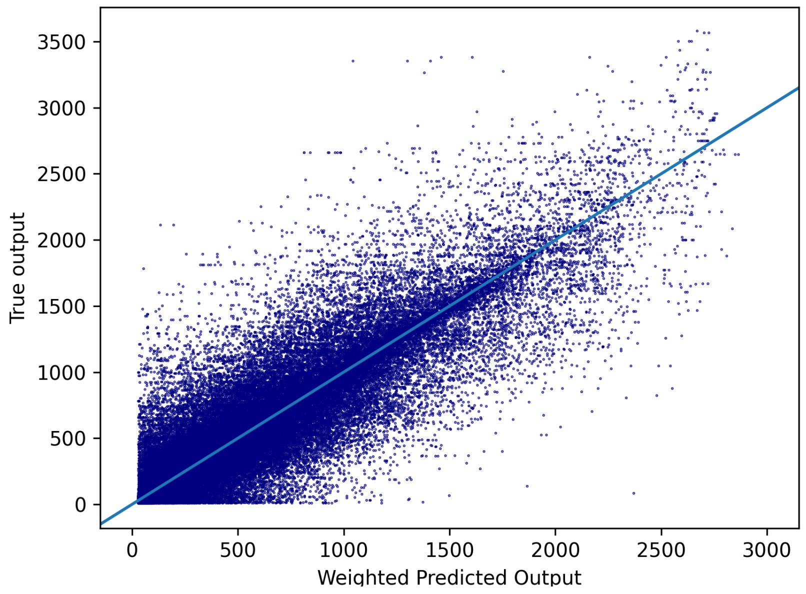


Figure 23 Predicted output vs true output for the GCNII - Buckets-e90 - extra data model on the test PrT dataset. Each point is a prediction for a single link, with both values being in veh/h. An optimal predicted would be along the light blue line. Notice that for true values above roughly 2500, the model constantly predicts around 2600.

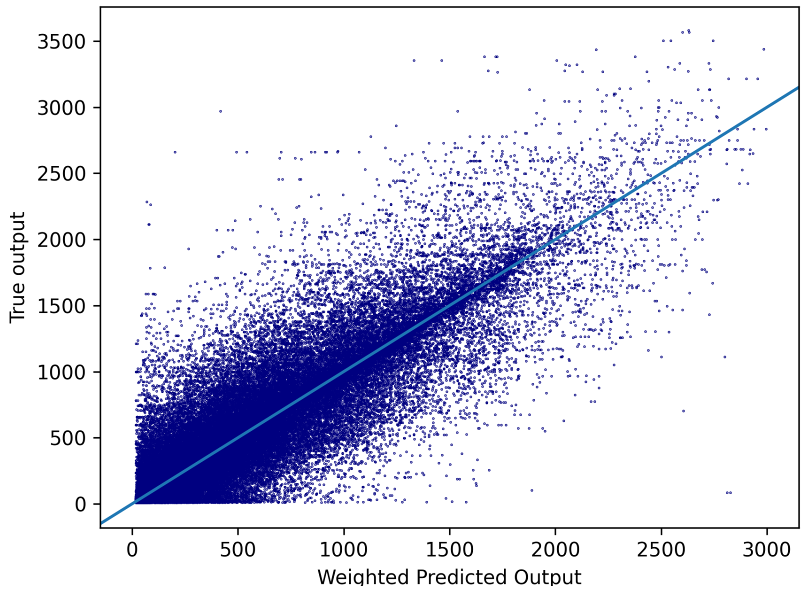


Figure 24 Predicted output vs true output for the GCNII - Buckets-nl54 - extra data model on the test PrT dataset. Each point is a prediction for a single link, with both values being in veh/h. An optimal predicted would be along the light blue line. Observe that for true values over 2500, the model still predicts proportionally high outputs.

loss function to train on the exact targets directly. However, none of the models in this configuration showed convergence, probably due to gradient issues by applying the softmax.

Interpretation and Findings

Comparing to the regression setting, the experiments here show that fine grained classification returns more consistent results with respect to the data processing, probably as fewer

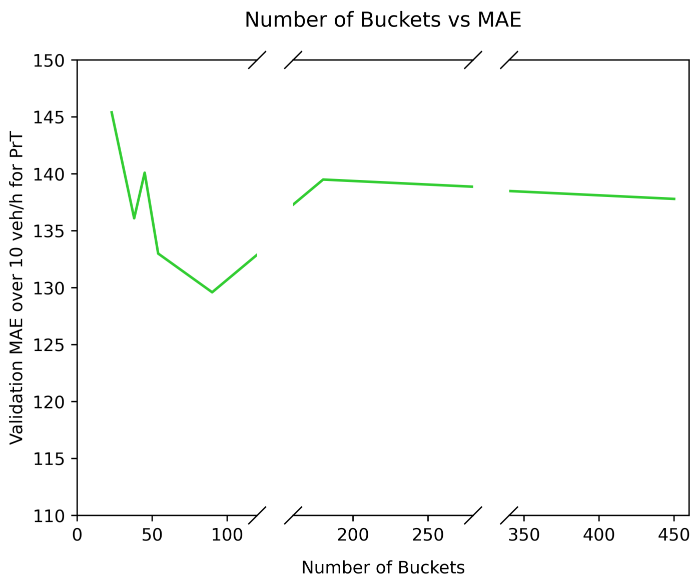


Figure 25 Validation $MAE_{\geq 10}$ in relation to the number of buckets on PrT classification for the GCNII model. For visualization purposes, large gaps without any new information are skipped.

processing steps are required. Such behaviour is especially desirable as it few configuration searches and reduces the overall model development time.

Looking at the equidistant bucket performance, having either too little or too many buckets results in worse performance, implying here that 90 buckets is optimal from the configurations tested. The results are also confirmed by the non-linear buckets, and it is important to note that there is not a major difference in metric performance between the two configurations.

Extra synthetic data has shown to make a significant improvement in the performance of the models, but is still not accurate enough to be used in real world scenarios. To confirm the results, the next experiments go into other GNN models, as well as the PuT outcomes, described in the next section.

Looking at the buckets, the equidistant 90 bucket has slightly worse $MAE_{\geq 10}$ performance, and has problems generalizing to larger values, specifically over 2500 veh/h. The effect comes probably from the high number of classes having few data points for training. On the other hand the non-linear 54 bucket model does not suffer from this, concluding that further experiments should use the 54 bucket configuration for better generalization.

7.3.2. Validating Non-Linear Buckets & Additional Synthetic Data - Results

To validate the previous results that non-linear buckets with extra data perform best, more GNNs are trained, both on the PrT and PuT tasks. The exact setup can be found in Chapter 6.3.2.

The results of the experiment runs can be found in Table 11. The best performing model is the

GCNII with extra data across both PrT and PuT, closely followed by the GCN+GAT model. All models experienced overfitting, seen as their training loss continuously reduced without an improvement in validation scores, but with GATv3 the effect was quite pronounced, resulting in poorer results. Both the GCNII and GCN+GAT with extra data outperform the basic models without extra data.

Table 11 Results of additional synthetic training data for fine grained classification on test datasets for both PrT and PuT, comparing different GNNs. GATv3 is in parentheses as it experienced overfitting.

Model	PrT - Test Set		PuT - Test Set	
	$R^2_{\geq 10}$	$MAE_{\geq 10}$	$R^2_{\geq 10}$	$MAE_{\geq 10}$
Mean Regressor	0.00	402.6	0.00	259.3
Random Forest	0.00	383.9	0.00	254.8
MLP	0.00	382.8	0.00	249.5
GCNII - Regression Best	0.84	135.4	0.61	141.2
GCNII - Buckets-nl54	0.79	136.4	0.62	136.9
GCNII - Buckets-nl54 - extra data	**0.85**	**118.5**	**0.64**	**131.8**
GATv3 - Buckets-nl54 - extra data	(0.78)	(144.9)	(0.54)	(152.0)
GCN+GAT - Buckets-nl54 - extra data	0.84	120.0	0.61	136.9

Looking at the means of the test datasets, it is 561.2 veh/h for the PrT dataset and 351.2 pers/h for PuT, excluding targets with less than 10 respective units per hour. In relation to the mean of the datasets, the current best model's $MAE_{\geq 10}$ is 21.1% error for PrT and 37.5% error for PuT. One of the assumptions of this thesis is that the relative error should be less than 10%. However, when examining the amount of samples individually within 10% relative error, only 35% of predictions achieve it for PrT and 22% for PuT. When including all predictions, analysed individually, within 10% relative error or within 50 respective units per hour, the amount of samples within this boundary becomes 54% for PrT and 41% for PuT.

To understand the root cause of the errors, various dataset parameters are examined. A major cause for large errors is found by comparing the graph size of the network to the respective $MAE_{\geq 10}$, depicted in Figure 26. A clear positive correlation is seen between the graph size and the error, meaning that the GNNs are not able to generalize well to larger networks. The effect is observed in all GNN models tested.

Interpretation and Findings

The improvements observed by increasing the training data is consistent with other GNN models, assuming they do not overfit on to the data. This implies that both more training data should be gathered for better predictions, as well as new regularization methods so that GNNs overfit less.

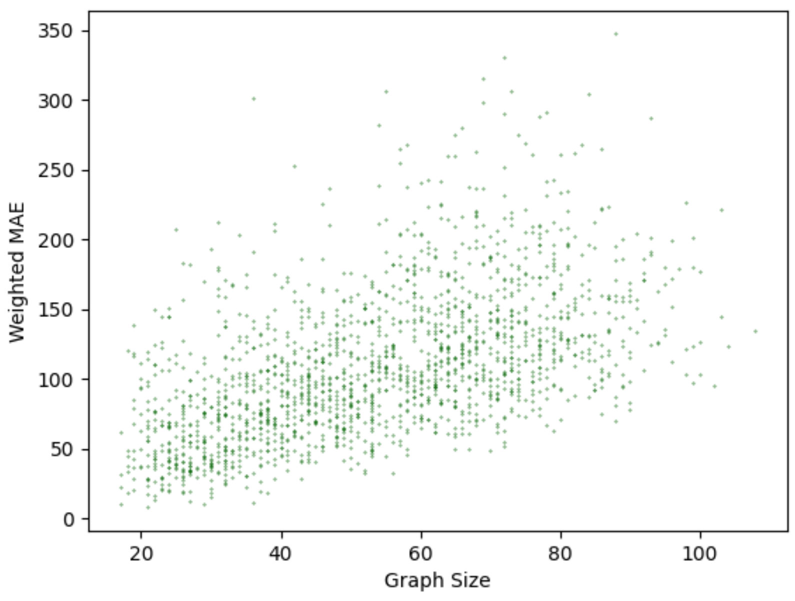


Figure 26 $MAE_{\geq 10}$ with respect to graph size, i.e. number of nodes for the trained GCN+GAT model on the PrT test dataset. Each point is the average absolute error of a single graph, with the errors coming from incorrect link transport usage prediction. Highly similar plots are observed for all GNNs.

Looking more closely at PuT, it consistently received worse predictions than PrT. The root cause of this cannot be currently determined, but a hypothesis is that there are less samples with at least 10 pers/h, meaning that the models do not receive enough training data.

The error analysis also reveals that the current best model is significantly over 10% error, the goal assumed for this thesis. One of the sources for the error is that GNNs perform worse for larger graphs, which is a major downside of the current state of the method. Additionally, in the GCN+GAT model, using first an artificial graph to connect nodes far from each other does not consistently resolve the problem. Further research should look into using GNNs for long distance processing.

7.3.3. Overall Fine Grained Classification Findings

The answer to whether fine grained classification is more accurate than regression, is yes. Especially with an extended training dataset, further performance improvements are reached where the specifications of the model do not make a difference. Additionally, classification is observed to be more stable with respect to model hyperparameters than regression. Unfortunately, the best models with extra training data do not reach errors less than 20%. A detailed analysis finds that an increasing network size correlates positively with an increased error.

8. Discussion

The following chapter consolidates the results of the previous experiments, by discussing which points reach their intended requirements, which do not, and hypothesize on their reasons. Additionally, insights for GNNs are presented, followed by the use cases for the existing components that work and finally an overview of the contributions.

8.1. Answer to Primary Research Question

The primary focus of this thesis is to analyse whether a Four Step Model surrogate can be built, especially by using a graph neural network. The following sections compare the functioning and non-functioning details, concluding with the answer to the primary research question.

8.1.1. Functioning Aspects

The experiments performed in this thesis show that many different aspects work as intended. Overall surrogate modelling procedure works for the Four Step Model, to varying degrees of success.

Focusing on the augmented data generation method is found to work well with deep learning, producing converging models that can generalize their outputs to new samples, as expected from existing literature (Nikolenko 2021). Additionally, a data transformation pipeline, and specifically the direct graph conversion technique are researched, proposed as well as implemented. By separating the PrT and PuT tasks into different models, it is possible to achieve stable convergence, a novel finding. On the basic classification scenario, the models achieve high prediction accuracy, a non-trivial result due to the low number of GNN tasks in such settings. The accuracy is validated on a qualitative analysis of the outputs, seeing that the model is able to understand vital transport aspects such as the congested network. Furthermore, the models have an over 50 time speed up in comparison to the PTV Visum baseline, are able to model all steps jointly and provide confidence intervals for outputs, original findings for transport planning supported by similar surrogate results in other simulation fields (Ogoke et al. 2020; Pfaff et al. 2020).

In further experiments, the state of the art model configuration is found. By using fine grained classification together with the expectation operator, it is possible to convert classification outputs of the GNNs into real values. Based on the bin search experiments, a heuristic is shown on developing a bin configuration, overlapping with existing results (Salman and Kecman 2012). Based on existing work, the current classification accuracy reaches that used for section control and congestion warning systems (Gerstenberger, Hösch, and Listl 2018).

Finally, by using a synthetic data generator, experiments show that it is possible to increase the training data without overfitting on existing data sources whilst improving model performance.

8.1.2. Non-functioning Aspects

On the other hand, a number of aspects are not functioning as needed for the surrogate model, as described in this section.

The primary downside are the errors, where even the best models do not reach below 120 $MAE_{\geq 10}$. This corresponds to an average of over 20%. An ideal error as determined by experts would be less than 10%, which is still far off.

Another downside of the model is that the task presented here is artificially simple. Both the demand and supply side are highly basic in the scenarios used here, for example, using simple socio-economic data and only two modes of transport. Additionally, the model in this thesis is never fine tuned on real world data which, even though is expected to be similar, is still important to validate that the transfer predictions work.

Looking more closely at the GNNs, there are still critical issues present. The most important is that the error correlates with graph sizes, which is a vital downside that needs to be addressed, as it has also been seen in other tasks (Alon and Yahav 2020; M. Chen et al. 2020). The underlying causes have been identified, such as over-squashing, though no solutions have been proposed. Additionally, the training time required for experiments grows exponentially with respect to prediction improvements, as highlighted in Figure 27. Finally, the depth of the GNN directly limits the maximum network size that can be used by the model, which in turn limits the practical usefulness of the surrogate.

8.1.3. Primary Research Question Conclusion

Recall, that the primary research question aims to understand whether a surrogate function can be built for the Four Step Model, focusing on solving the sequential assumptions on the travelers decision making, the deterministic outputs, the manual calibration and the long run times of simulations.

Observing the results of the experiments, GNNs as a surrogate model are able to solve the four downsides of the Four Step Model, but do not reach the accuracy needed for practical use. The thesis provides a foundation on which more powerful and accurate surrogate models can be built. The aim of the surrogate is to have a model which reaches within 10% accuracy, as assumed in this thesis.

Looking at the reasons as to why the model does not reach the needed accuracy, there are two main points. The first is the need for more data, whilst the second is to have more advanced GNN models without performance degradation from large graphs as input.

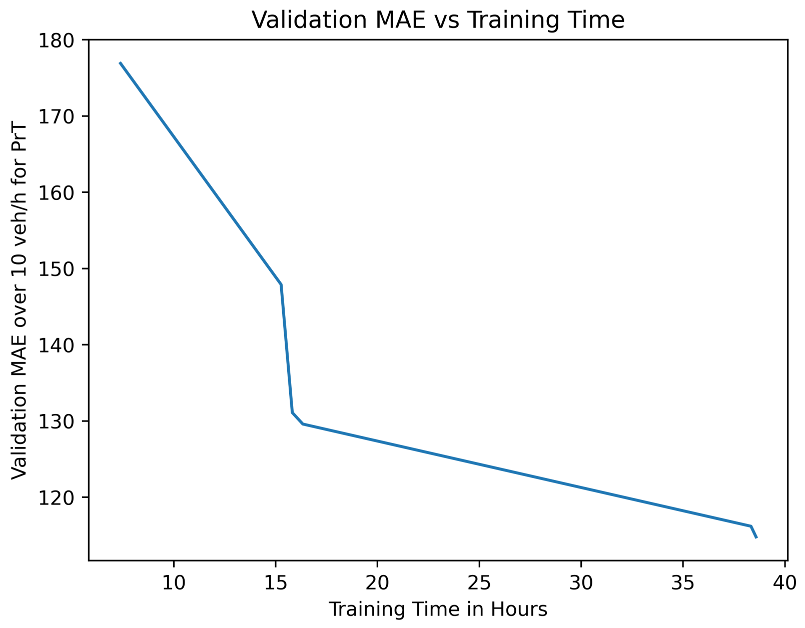


Figure 27 Validation $MAE_{\geq 10}$ for key PrT GCNII experiments in relation to the training time in hours. Notice the logarithmic improvements of MAE.

The concluding answer to the primary research question "Can a proof of concept deep learning surrogate be built for the 4 Step Model for networks with a maximum of 80 nodes?" is yes, but not yet to an accuracy usable in practise.

8.2. Insights for Graph Neural Networks

A number of insights are also found for graph neural networks in this thesis, presented in the next few paragraphs.

The general problem formulation, which combines GNN transfer learning together with node classification for entire graphs, has not been researched much in literature before (Dwivedi et al. 2020). The results of the thesis find that GNNs can be used for these tasks.

Regarding the models itself, it is found that it is possible to develop deep GATs, using the GATv3 formulation, first proposed by the author of this thesis. The resulting model is capable of learning more complex patterns that the shallow counterpart, and thus be used for larger graphs. However, larger graphs still are shown to worsen performance, with existing literature pointing to over-squashing and over-smoothing, as highlighted in the models chapter. Combining GCN and GAT, and using artificial longer connection graphs in the initial layers, is shown to not alleviate the problem, which is a critical understanding towards finding a way to overcome the bottlneck of GNNs. Overall, the results of the models show that it is possible to make stable, deep GNNs, but the issue of working on large graphs is not solved.

Additionally, the experiments of this thesis show that deeper GNNs perform better than their shallow counterparts, which has also not been observed often in existing literature (M. Chen et al. 2020), primarily due to subpar model formulations and a lack of data. By adding in synthetic data, the GNN models perform better, which is in line with observations in other application fields of deep neural networks (Nikolenko 2021).

An extensive comparison of regression and classification is performed, together with a conversion technique. It is empirically found that classification, or more formally cross entropy, is more stable than the regression loss, MSE, with respect to the hyperparameters, also corresponding to existing literature (Rady 2008; Salman and Kecman 2012). This finding is important so that future work in transport planning focuses on classification, and finding ways to make it more accurate. Trying to combine the two approaches together in hierarchical regression also is shown not to produce more accurate results, with no similar procedures found in existing literature for deep learning. Finally, first converting classification outputs using the expectation operator and then training on MSE has shown not to converge, which is to be expected due to the small magnitude of the gradients.

8.3. Use Cases for Existing Surrogate Models

Even though the best surrogate Four Step Model found in this thesis does not reach practical accuracies, there are possible applications in the transport planning field for the models, described in the following section.

The first potential application is to use the less accurate model as the initial configuration for trip generation, mode choice and trip assignment in the Four Step Model. The configurations can be easily derived by performing aggregations on the outputs of the surrogate. Even though the exact values are not perfect, the resulting model could potentially require less iteration steps to reach convergence.

The models presented here achieve 95% accuracy for determining links with no transport usage, which could be used to optimize trip assignment algorithms by shrinking the search space. With a high accuracy level, it is thus possible that existing GNN models can have similar performance on larger, more realistic networks, meaning that large Four Step Models could reduce their runtimes.

The models accuracy seen here allows the estimation of coarsely bucketed congestion levels, as defined by (Gerstenberger, Hösch, and Listl 2018). The congestion level application is perhaps the closest working aspect for the existing surrogate models of this thesis.

Overall, the current use cases for the models are based either on highly specific use cases, or to accelerate the convergence of the existing simulation models. Further research ideas are presented in the final chapter.

9. Conclusion

Coming to the conclusion of the thesis, a brief overview is presented, a summary of the contributions is shown, and finally the potential future research directions are described.

9.1. Overview

Starting off with a summary of the work conducted, a literature review is provided on the downsides of the Four Step Model, identifying that a deep learning surrogate can potentially solve them. Based on the idea, a formal definition of the task is presented. Coming to the modelling part, first surrogate model baselines are presented, based on classical machine learning methods. Next, the GCN and GCNII are introduced and implemented. Additionally, GATv3 is proposed and it is shown to be the first GAT model in literature to achieve large depths, as well as the GCN+GAT model attempting to solve the over-squashing problem. For the experiments, the metrics F1, $MAE_{\geq 10}$ and $R^2_{\geq 10}$ are defined for the surrogate development in transport planning. Next, augmented and synthetic data generation procedures are implemented for easy dataset creation, which can be recreated in academic settings. To the best of the author's knowledge, the augmented and synthetic datasets proposed and generated in this thesis are the largest GNN surrogate modelling datasets currently available. In general, the dataset used here is one of the largest for GNN training for node classification or node regression (Dwivedi et al. 2020; Errica et al. 2020). Based on preliminary experiments, a stable graph processing strategy is found that first transforms edges into links and then separates PrT and PuT into subtasks. Based on the experimental results, it is demonstrated that a basic surrogate model can be built that solves a number of major problems of the Four Step Model, with consistent convergence and the ability to model critical transport aspects, such as the congested network. In more complex experiments, fine grained classification is found to be more stable than regression when trying to obtain precise values. When combining classification and regression in a novel formulation, the results of the model do not improve. However, by using the double expectation operator, it is shown to be possible to convert classification outputs into real values to improve predictions. In the fine grained classification experiments, a stable binning strategy is identified and in the last experiments it is observed that more training data improves predictions when attempting to predict real values.

The answer to the primary research question, looking at whether a proof of concept surrogate can be built, is yes, but currently with limited accuracy not usable in practise yet. A configuration of data generation, transformation and GNN models is found which results in stable convergence, based on augmented data, direct graph conversion as well as deep GNNs with at least 20 layers. Additionally, it is found that GNNs do not generalize well to larger graphs, probably due to the over-squashing problems identified in existing literature. The findings highlight that GNNs are still a maturing technology, and require a critical breakthrough for larger graphs.

The key contributions of the thesis are as following:

1. The first surrogate platform for the Four Step Model is proposed. The contributions here are identifying the gap in the literature, formalizing the problem, defining the domain relevant assumptions and developing the novel pipeline. The results of this thesis open up the strategic transport planning problem to deep learning research.
2. Two novel data generation algorithms, based on synthetic and augmented data, are developed specifically within the scope of this thesis for transport planning. The resulting dataset is the largest overall dataset for surrogate modelling of graphs, and one of the first in literature, for any graph task, to require deep models to be modelled correctly. Finally, a modular data processing pipeline is implemented, and two graph transformation strategies are taken to ensure deep learning models are able to use the input data.
3. In the surrogate model section, two novel models are proposed, the GATv3 as well as the GCN+GAT. GATv3 is the first GAT to achieve large depths in literature, which is a critical step not only for this specific task, but for any graph problem requiring that nodes at different ends of the graph interact. GCN+GAT attempts to resolve the over-squashing problem, a critical open issue for all graph neural networks. Finally, the GCN, GCNII, MLP, Random Forest and null models are implemented as well.
4. All experiments and training runs conducted are completely novel. The analysis and results show a large collection of experiments, allowing future research to focus on the most promising direction.

To conclude, the main point of the thesis is that it is possible to build a surrogate model for the Four Step Model, but the required accuracy is not yet achieved. Further development should go into models that handle large networks better and, more importantly, the creation of larger as well as more realistic open datasets. In the final section, future research opportunities are presented. Overall, this thesis provides a platform for future research on surrogate models for the Four Step Model, as well as a new task that graph neural networks need to solve.

9.2. Future Research

Based on the results of the experiments in this thesis, a number of possible future research directions can be taken. The final section first looks at how future research should look into the GNN surrogate for the Four Step Model, next at how GNNs can be used in transport planning and finally into the research of GNNs.

9.2.1. Graph Neural Network Surrogate for the Four Step Model

For the surrogate model, future research should be aimed at increasing the accuracy. Based on the results of the previous chapter, the models should be deeper and more expressive. More data should also be employed, probably over 100000 samples for smaller networks. Data generation can be done both using synthetic and augmented data sources, though the final models should be evaluated on real world data.

The GNN models should probably be based on a classification model, using non-linear buckets to ensure enough samples in each category and a conversion into real values using the double expectation operator defined in the experiments chapter. As GNNs are still developing at a rapid pace at the time of writing, the state of the art should be examined and used. However, the best configuration here is shown to use GraphNorm, deep GCNII layers, masking of low value targets, dropout and residual final layers.

Looking at the Four Step Model for the proof of concept, the model used as the target is basic. Future work also needs to employ more realistic 4SM, which also require more data to be generated. Additionally, using a joint PrT and PuT assignment in the 4SM could also potentially stabilize joint training of the GNN surrogates. Finally, using various different 4SM during the data generation stage is a key step for the surrogate to be useful across different scenarios.

9.2.2. Application of Graph Neural Networks in Transport Planning

Predicting all four stages together by a surrogate are not yet accurate enough, however in the final section, a number of potential fields for GNNs in general transport planning are found for future research to examine.

Looking at the individual steps of the Four Step Model, it may be possible to replace the traffic assignment step with a graph neural network. The increased speed would allow more experiments to be conducted, and the assignment step to be automatically calibrated via backpropagation. Additionally, as GNNs are differentiable, it may be possible to propagate the prediction error to previous steps, making automatic calibration over multiple modelling steps possible (X. Wu et al. 2018).

Finally, many problems in transport can be formulated as transfer learning node classification tasks, and even more as general problems for graphs. Based on both existing work and the result of this thesis, GNNs can already be used in smaller tasks to directly predict values on networks. By taking the underlying network explicitly into consideration, higher accuracies could be achieved in comparison to traditional machine learning methods. Additionally, GNNs could potentially be used in operations research tasks to approximate the optimum with much faster convergence (Cappart et al. 2021), for example for the fleet setup. Future research should examine how GNNs can solve other problems in transport planning.

9.2.3. Graph Neural Network Research

Looking more closely at GNNs, there is still a number of significant issues, with the most pressing for transport planning presented in the next paragraphs.

The first issue is that they are shown to have performance degradation due to over-smoothing and over-squashing when building deeper models for larger graphs. Over-smoothing has been primarily solved with the tricks shown in GCNII, but over-squashing remains a critical issue that needs to be addressed. Especially with larger graphs, a direction would be a system to stop large error propagation, potentially similar to the training function of large language models in natural language processing (Brown et al. 2020).

Next, GNNs are currently using a fixed depth, meaning that the theoretical limit on the network size that they can process is directly proportional to their depth and fixed for a model. Future research should examine how to break this barrier, for example by using similar ideas as recurrent neural networks (Hochreiter and Schmidhuber 1997). Finally, more experiments in GNN node classification for transfer learning should be done. Specifically, a reduction of data usage for training and improved convergence are a major issue.

Overall, the fields of both transport planning surrogates and graph neural networks require more research, with a number of topics open for analysis. The most pressing issues are focusing on getting GNNs to work on arbitrarily large graphs, and constructing a large open dataset for transport planning.

A. Data Transformation Module Categories

A number of further data processing modules are created to enable easy experimentation. Altogether 29 modules are implemented as part of this thesis and the overall categories are listed below. None of the modules are completely novel, however, the combination and specific implementation in the modular framework within this thesis is new.

Visualizations

A number of visualization tools are developed, including histogram and error plotting. Individual graphs can also be visualized easily, to understand how the individual steps are affecting the data. Even though visualizations are often seen as optional features, they help find bugs quicker, speeding up the experimentation process.

Attribute Processing

A number of modules allow automatic attribute removal as well as inserting additional attributes with default values.

Attribute Normalizing and Standardizing

Further modules add in normalizing and standardizing functionality. This is especially important as neural networks require that all data be around the range of -1 to 1. Additionally, spacial aspects such as coordinates and directions need to be preserved and provided as input.

Bucketing

A number of bucketing options are implemented, to transform real values into discrete numbers, by identifying in which bucket they fall. This part is important for classification.

Scaling

For regression tasks, modules allowing linear and logarithmic scaling are introduced.

Graph Processing

Especially when using baseline machine learning methods, abnormally large graphs, that are in the data due to edge cases, need to be filtered out. Additionally, transforming the graphs into flat vectors for simpler machine learning methods is added.

PyTorch Tensors

In this thesis, PyTorch is used as the deep learning framework. For proper input usage, the data needs to be finally transformed into PyTorch tensors from Numpy arrays, and sent to the proper processing device, which is usually the GPU.

Cache Manager   As the processing time often takes up to 15 minutes per run, a cache manager is implemented. The cache manager automatically identifies the data processing steps by looking at the steps' hash values, and if already performed before, retrieves the processed data from a storage. This enables much quicker experimentation.

Bibliography

Abrahamsson, T. and L. Lundqvist (Feb. 1999). “Formulation and Estimation of Combined Network Equilibrium Models with Applications to Stockholm”. In: *Transportation Science* 33.1, pp. 80–100. DOI: 10.1287/trsc.33.1.80.

Ali Safwat, K Nabil and Thomas L Magnanti (1988). “A combined trip generation, trip distribution, modal split, and trip assignment model”. In: *Transportation Science* 22.1, pp. 14–30. DOI: 10.1016/0041-1647(78)90065-5.

Alon, Uri and Eran Yahav (2020). “On the Bottleneck of Graph Neural Networks and its Practical Implications”. In: *8th International Conference on Learning Representations (ICLR) 2020*. URL: https://arxiv.org/abs/2006.05205.

Antoniou, Constantinos, Dimitrios Efthymiou, and Emmanouil Chaniotakis (2019). *Demand for Emerging Transportation Systems: Modeling Adoption, Satisfaction, and Mobility Patterns*. Amsterdam: Elsevier. ISBN: 9780128150191.

Bahar, Parnia, Nikita Makarov, and Hermann Ney (2020). “Investigation of transformer-based latent attention models for neural machine translation”. In: *14th Conference of the Association for Machine Translation in the Americas (AMTA 2020)*, pp. 7–20. URL: https://www-i6.informatik.rwth-aachen.de/publications/download/1150/Bahar-AMTA%5C%20-2020.pdf.

Bahar, Parnia, Nikita Makarov, Albert Zeyer, et al. (2020). “Exploring a zero-order direct HMM based on latent attention for automatic speech recognition”. In: *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 7854–7858. DOI: 10.1109/ICASSP40776.2020.9054545.

Bai, Jiyang, Yuxiang Ren, and Jiawei Zhang (2021). “Ripple walk training: A subgraph-based training framework for large and deep graph neural network”. In: *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. URL: https://arxiv.org/abs/2002.07206.

Belkin, Mikhail et al. (2019). “Reconciling modern machine-learning practice and the classical bias–variance trade-off”. In: *National Academy of Sciences* 116.32, pp. 15849–15854. URL: https://arxiv.org/abs/1812.11118.

Benoit, Kenneth (2011). “Linear regression models with logarithmic transformations”. In: *London School of Economics, London* 22.1, pp. 23–36. URL: https://kenbenoit.net/assets/courses/me104/logmodels2.pdf.

Bergstra, James and Yoshua Bengio (2012). “Random search for hyper-parameter optimization.” In: *Journal of Machine Learning Research* 13.2. URL: https://dl.acm.org/doi/10.5555/2188385.2188395.

Bleeg, James (Sept. 2020). “A Graph Neural Network Surrogate Model for the Prediction of Turbine Interaction Loss”. In: *Journal of Physics: Conference Series* 1618, p. 062054. DOI: 10.1088/1742-6596/1618/6/062054.

Boyce, David E, Yu-Fang Zhang, and Mary R Lupa (1994). “Introducing" feedback" into four-step travel forecasting procedure versus equilibrium solution of combined model”. In: *Transportation Research Record* 1443, p. 65. URL: https://trid.trb.org/view/414453.

Brinckerhoff, Parsons (2010). “Base Year Update and Validation of the NYMTC”. In: *New York Metropolitan Transportation Council (NYMTC), prepared by Parsons Brinckerhoff*. URL: https://www.nymtc.org/DATA-AND-MODELING/New-York-Best-Practice-Model/Model-Update.

Brody, Shaked, Uri Alon, and Eran Yahav (2021). “How Attentive are Graph Attention Networks?” In: *Preprint*. URL: https://arxiv.org/abs/2105.14491.

Brown, Tom B et al. (2020). “Language models are few-shot learners”. In: *Preprint*. URL: https://arxiv.org/abs/2005.14165.

Cai, Tianle et al. (2021). “Graphnorm: A principled approach to accelerating graph neural network training”. In: *International Conference on Machine Learning*, pp. 1204–1215. URL: https://arxiv.org/abs/2009.03294.

Cappart, Quentin et al. (2021). “Combinatorial optimization and reasoning with graph neural networks”. In: *Preprint*. URL: https://arxiv.org/abs/2102.09544.

Chen, Ming et al. (2020). “Simple and deep graph convolutional networks”. In: *International Conference on Machine Learning*, pp. 1725–1735. URL: https://arxiv.org/abs/2007.02133.

Cheng, Long et al. (Jan. 2019). “Applying a random forest method approach to model travel mode choice behavior”. In: *Travel Behaviour and Society* 14, pp. 1–10. DOI: 10.1016/j.tbs.2018.09.002.

Chetty, Raj, Nathaniel Hendren, et al. (2015). “The impacts of neighborhoods on intergenerational mobility: Childhood exposure effects and county-level estimates”. In: *Harvard University and NBER*, pp. 1–144. URL: https://opportunityinsights.org/wp-content/uploads/2018/10/nbhds_paper.pdf.

Cogo, Emir et al. (Oct. 2019). “Survey of integrability of procedural modeling techniques for generating a complete city”. In: *2019 XXVII International Conference on Information, Communication and Automation Technologies (ICAT)*. IEEE. DOI: 10.1109/icat47117.2019.8939012.

D’Ambrosio, Antonio and Valerio A Tutore (2011). “Conditional classification trees by weighting the gini impurity measure”. In: *New perspectives in statistical modeling and data analysis*, pp. 273–280. DOI: 10.1007/978-3-642-11363-5_31.

Das, Kalyan, Jiming Jiang, and JNK Rao (2004). “Mean squared error of empirical predictor”. In: *The Annals of Statistics* 32.2, pp. 818–840. URL: https://arxiv.org/abs/math/0406455.

Dios Ortúzar, Juan de and Luis G Willumsen (2011). *Modelling transport*. Oxford: Wiley-Blackwell. ISBN: 978-0-470-76039-0.

Dwivedi, Vijay Prakash et al. (2020). “Benchmarking Graph Neural Networks”. In: *TSP* 12, pp. 50–500. URL: https://arxiv.org/abs/2003.00982.

Efron, Bradley and Trevor Hastie (2016). *Computer age statistical inference*. Vol. 5. New York, NY: Cambridge University Press, p. 264. ISBN: 978-1107149892.

Errica, Federico et al. (2020). “A Fair Comparison of Graph Neural Networks for Graph Classification”. In: *8th International Conference on Learning Representations (ICLR 2020)*. URL: https://arxiv.org/abs/1912.09893.

Feng, Xuesong, Baohua Mao, and Quanxin Sun (2010). “Comparative Evaluations on Feedback Solutions to the Feedback Process of a New Model for Urban Transport Study”. In: *International Journal of Urban Sciences* 14.2, pp. 164–175. DOI: 10.1080/12265934.2010.9693674.

Fey, Matthias and Jan E. Lenssen (2019). “Fast Graph Representation Learning with PyTorch Geometric”. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. URL: https://arxiv.org/abs/1903.02428.

Freitas, Lucas Meyer de et al. (2019). “Modelling intermodal travel in Switzerland: A recursive logit approach”. In: *Transportation Research Part A: Policy and Practice* 119, pp. 200–213. DOI: 10.1016/j.tra.2018.11.009.

Friedman, Jerome H (2017). *The Elements of Statistical Learning: Data mining, Inference, and Prediction*. New York: Springer.

Fürnkranz, Johannes et al. (2011). “Mean Squared Error”. In: *Encyclopedia of Machine Learning*. Springer US, pp. 653–653. DOI: 10.1007/978-0-387-30164-8_528.

Gerstenberger, Marcus, Michael Hösch, and Gerhard Listl (2018). *Überarbeitung und Aktualisierung des Merkblattes für die Ausstattung von Verkehrsrechner- und Unterzentralen (MARZ 1999)*. Bundesanstalt für Straßenwesen (BASt). URL: https://trid.trb.org/view/1567477.

Goel, Shivendra and Ashok K Sinha (2015). “ANFIS Based Trip Distribution model for Delhi Urban Area”. In: *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 453–457. URL: https://ieeexplore.ieee.org/document/7100291.

Grunitzki, Ricardo and Ana L. C. Bazzan (Oct. 2017). “Comparing Two Multiagent Reinforcement Learning Approaches for the Traffic Assignment Problem”. In: *2017 Brazilian Conference on Intelligent Systems (BRACIS)*. IEEE. DOI: 10.1109/bracis.2017.19.

Hagberg, Aric, Pieter Swart, and Daniel S Chult (2008). *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States). URL: https://www.osti.gov/biblio/960616-exploring-network-structure-dynamics-function-using-networkx.

Hagenauer, Julian and Marco Helbich (July 2017). “A comparative study of machine learning classifiers for modeling travel mode choice”. In: *Expert Systems with Applications* 78, pp. 273–282. DOI: 10.1016/j.eswa.2017.01.057.

Harary, Frank and Robert Z. Norman (May 1960). “Some properties of line digraphs”. In: *Rendiconti del Circolo Matematico di Palermo* 9.2, pp. 161–168. DOI: 10.1007/bf02854581.

Harris, Charles R. et al. (Sept. 2020). “Array programming with NumPy”. In: 585.7825, pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

Hasan, Mohamad K and Hussain M Dashti (2007). “A multiclass simultaneous transportation equilibrium model”. In: *Networks and spatial economics* 7.3, pp. 197–211. DOI: 10.1007/s11067-006-9014-3.

He, Kaiming et al. (2016). “Deep residual learning for image recognition”. In: *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778. URL: `https://arxiv.org/abs/1512.03385`.

He, Xin, Kaiyong Zhao, and Xiaowen Chu (2021). “AutoML: A Survey of the State-of-the-Art”. In: *Knowledge-Based Systems* 212, p. 106622. URL: `https://arxiv.org/abs/1908.00709`.

Ho, Tin Kam (1995). “Random decision forests”. In: *3rd international conference on document analysis and recognition*. Vol. 1, pp. 278–282. DOI: `10.1109/ICDAR.1995.598994`.

Hochreiter, Sepp and Jürgen Schmidhuber (1997). “Long short-term memory”. In: *Neural computation* 9.8, pp. 1735–1780. DOI: `10.1162/neco.1997.9.8.1735`.

Hong, Jungyeol et al. (2019). “Exploring the topological characteristics of complex public transportation networks: Focus on Variations in both single and integrated systems in the Seoul metropolitan area”. In: *Sustainability* 11.19, p. 5404. DOI: `10.3390/su11195404`.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). “Multilayer feedforward networks are universal approximators”. In: *Neural networks* 2.5, pp. 359–366. DOI: `10.1016/0893-6080(89)90020-8`.

Iliashenko, Oksana, Victoria Iliashenko, and Ekaterina Lukyanchenko (2021). “Big Data in Transport Modelling and Planning”. In: *Transportation Research Procedia* 54, pp. 900–908. DOI: `10.1016/j.trpro.2021.02.145`.

Jiang, Weiwei and Jiayun Luo (2021). “Graph neural network for traffic forecasting: A survey”. In: *Preprint*. URL: `https://arxiv.org/abs/2101.11174`.

Kaewwichian, Patiphan (Oct. 2019). “Car ownership demand modeling using machine learning: Decision Trees and Neural Networks”. In: *International Journal of GEOMATE* 17.62. DOI: `10.21660/2019.62.94618`.

Kelvin, Lin Ziwen and Bhojan Anand (Sept. 2020). “Procedural Generation of Roads with Conditional Generative Adversarial Networks”. In: *2020 IEEE Sixth International Conference on Multimedia Big Data (BigMM)*. IEEE. DOI: `10.1109/bigmm50055.2020.00048`.

Kiefer, Jack and Jacob Wolfowitz (1952). “Stochastic estimation of the maximum of a regression function”. In: *The Annals of Mathematical Statistics*, pp. 462–466. DOI: `10.1214/aoms/1177729392`.

Kingma, Diederik P. and Jimmy Ba (2015). “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. URL: `http://arxiv.org/abs/1412.6980`.

Kipf, Thomas N and Max Welling (2017). “Semi-supervised classification with graph convolutional networks”. In: *5th International Conference on Learning Representations, ICLR 2017*. URL: `https://arxiv.org/abs/1609.02907`.

Kompil, Mert and H Murat Celik (2013). “Modelling trip distribution with fuzzy and genetic fuzzy systems”. In: *Transportation Planning and Technology* 36.2, pp. 170–200. DOI: `10.1080/03081060.2013.7705??`

Kunst, Alexander (Sept. 2021). *Car ownership in Germany 2021*. URL: `https://www.statista.com/forecasts/998697/car-ownership-in-germany`.

Li, Dehui et al. (2012). “Transportation Demand Modeling Method Research—A Case Study of Chengdu”. In: *Civil Engineering and Urban Planning 2012*, pp. 799–807. DOI: 10.1016/j.trc.2019.01.030.

Li, Qimai, Zhichao Han, and Xiao-Ming Wu (2018). “Deeper insights into graph convolutional networks for semi-supervised learning”. In: *Thirty-Second AAAI conference on Artificial Intelligence*. URL: https://arxiv.org/abs/1801.07606.

Manheim, Marvin (1979). *Fundamentals of Transportation systems analysis; Volume 1: Basic concepts*. Cambridge, Mass: MIT Press. ISBN: 9780262131292.

Manibardo, Eric L., Ibai Laña, and Javier Del Ser (2021). “Deep Learning for Road Traffic Forecasting: Does It Make a Difference?” In: *IEEE Transactions on Intelligent Transportation Systems (Early Access)*, pp. 1–25. DOI: 10.1109/TITS.2021.3083957.

Manzo, Stefano, Otto Anker Nielsen, and Carlo Giacomo Prato (Jan. 2014). “Effects of Uncertainty in Speed–Flow Curve Parameters on a Large-Scale Model”. In: *Transportation Research Record: Journal of the Transportation Research Board* 2429.1, pp. 30–37. DOI: 10.3141/2429-04.

– (Feb. 2015). “How uncertainty in input and parameters influences transport model :output A four-stage model case-study”. In: *Transport Policy* 38, pp. 64–72. DOI: 10.1016/j.tranpol.2014.12.004.

McNally, Michael G. and Craig R. Rindt (Sept. 2007a). “The Activity-Based Approach”. In: *Handbook of Transport Modelling*. Emerald Group Publishing Limited, pp. 55–73. DOI: 10.1108/9780857245670-004.

– (Sept. 2007b). “The Four-step Model”. In: *Handbook of Transport Modelling*. Emerald Group Publishing Limited, pp. 35–53. DOI: 10.1108/9780857245670-004.

Miller, John W, Rod Goodman, and Padhraic Smyth (1993). “On loss functions which minimize to conditional expected values and posterior probabilities”. In: *IEEE Transactions on Information Theory* 39.4, pp. 1404–1408. DOI: 10.1109/18.243457.

Mladenovic, Milos and Aleksandar Trifunovic (2014). “The shortcomings of the conventional four step travel demand forecasting process”. In: *Journal of Road and Traffic Engineering* 60.1, pp. 5–12. URL: https://www.researchgate.net/publication/263423775_The_Shortcomings_of_the_Conventional_Four_Step_Travel_Demand_Forecasting_Process.

Mockus, Jonas (1989). *Bayesian Approach to Global Optimization : Theory and Applications*. Dordrecht: Springer Netherlands. ISBN: 9789401068987.

Moore, Edward F (1959). “The shortest path through a maze”. In: *Proc. Int. Symp. Switching Theory, 1959*, pp. 285–292.

Najmi, Ali et al. (2020). “Calibration of Large-scale Transport Planning Models: A Structured Approach”. In: *Transportation* 47.4, pp. 1867–1905. DOI: 10.1007/s11116-019-10018-6.

Navarro-Ligero, Miguel L., Julio A. Soria-Lara, and Luis Miguel Valenzuela-Montes (Aug. 2019). “A Heuristic Approach for Exploring Uncertainties in Transport Planning Research”. In: *Planning Theory & Practice* 20.4, pp. 537–554. DOI: 10.1080/14649357.2019.1648851.

Nielsen, Michael A. (2015). *Neural Networks and Deep Learning*. URL: http://www.neuralnetworksanddeeplearning.com/chap2.html.

Nikolenko, Sergey (2021). “Synthetic data for deep learning”. In:

Nwankpa, Chigozie Enyinna et al. (2021). “Activation functions: comparison of trends in practice and research for deep learning”. In: *2nd International Conference on Computational Sciences and Technology*, pp. 124–133.

Ogoke, Francis et al. (2020). “Graph convolutional neural networks for body force prediction”. In: *Preprint*. URL: https://arxiv.org/abs/2012.02232.

Omrani, Reza and Lina Kattan (Jan. 2012). “Demand and Supply Calibration of Dynamic Traffic Assignment Models”. In: *Transportation Research Record: Journal of the Transportation Research Board* 2283.1, pp. 100–112. DOI: 10.3141/2283-11.

OpenStreetMap contributors (2017). *Planet dump retrieved from https://planet.osm.org*. URL: https://www.openstreetmap.org.

Parish, Yoav I. H. and Pascal Müller (2001). “Procedural modeling of cities”. In: *28th annual conference on Computer graphics and interactive techniques - SIGGRAPH ’01*. ACM Press. DOI: 10.1145/383259.383292.

Paszke, Adam et al. (2019). “PyTorch: An Imperative Style, High-Performance Deep Learning Library”. In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830. URL: http://jmlr.org/papers/v12/pedregosa11a.html.

Pfaff, Tobias et al. (2020). “Learning Mesh-Based Simulation with Graph Networks”. In: *8th International Conference on Learning Representations (ICLR) 2020*. URL: https://arxiv.org/abs/2010.03409.

Pineda-Jaramillo, Juan D. (Oct. 2019). “A review of Machine Learning (ML) algorithms used for modeling travel mode choice”. In: *DYNA* 86.211, pp. 32–41. DOI: 10.15446/dyna.v86n211.79743.

Pitombo, Cira Souza, Andreza Dornelas de Souza, and Anabele Lindner (Apr. 2017). “Comparing decision tree algorithms to estimate intercity trip distribution”. In: *Transportation Research Part C: Emerging Technologies* 77, pp. 16–32. DOI: 10.1016/j.trc.2017.01.009.

Pourebrahim, Nastaran et al. (Sept. 2019). “Trip distribution modeling with Twitter data”. In: *Computers, Environment and Urban Systems* 77, p. 101354. DOI: 10.1016/j.compenvurbsys.2019.101354.

Pouyanfar, Samira et al. (Jan. 2019). “A Survey on Deep Learning”. In: *ACM Computing Surveys* 51.5, pp. 1–36. DOI: 10.1145/3234150.

PTV Group (2021). *Visum*. URL: https://www.ptvgroup.com/en/solutions/products/ptv-visum/.

Qin, Xiao and Hani S. Mahmassani (Jan. 2004). “Adaptive Calibration of Dynamic Speed-Density Relations for Online Network Traffic Estimation and Prediction Applications”. In:

Transportation Research Record: Journal of the Transportation Research Board 1876.1, pp. 82–89. DOI: 10.3141/1876-09.

Rady, Hussein (2008). “Classification of multilayer neural networks using cross entropy and mean square errors”. In: *El Shorouk, Journal of the ACM (Advances in Computer Science)*. DOI: 10.21608/ASC.2008.148466.

Ramadan, Abdelrahman et al. (June 2020). “Traffic Forecasting using Temporal Line Graph Convolutional Network: Case Study”. In: *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*. IEEE. DOI: 10.1109/icc40277.2020.9149233.

Rasouli, Soora and Harry Timmermans (Jan. 2012). “Uncertainty in travel demand forecasting models: literature review and research agenda”. In: *Transportation Letters* 4.1, pp. 55–73. DOI: 10.3328/tl.2012.04.01.55-73.

Reitermanova, Zuzana (2010). “Data splitting”. In: *WDS*. Vol. 10, pp. 31–36. URL: https://www.semanticscholar.org/paper/Data-Splitting-Reitermanov%5C%C3%5C%A1/3804cb787031aacd41c2de320bc4ddad637238a3.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). “Learning representations by back-propagating errors”. In: *nature* 323.6088, pp. 533–536. DOI: 10.1038/323533a0.

Saleh, S M et al. (Feb. 2021). “Trip generation and attraction model and forecasting using machine learning methods”. In: *IOP Conference Series: Materials Science and Engineering* 1087.1, p. 012021. DOI: 10.1088/1757-899x/1087/1/012021.

Salman, Raied and Vojislav Kecman (Mar. 2012). “Regression as classification”. In: *2012 IEEE Southeastcon*. IEEE. DOI: 10.1109/secon.2012.6196887.

Sasaki, Yutaka et al. (2007). “The truth of the f-measure. 2007”. In: URL: https://www.cs.odu%20edu/%5C~%7B%7D%20mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf.

Shan, Songqing and G. Gary Wang (Aug. 2009). “Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions”. In: *Structural and Multidisciplinary Optimization* 41.2, pp. 219–241. DOI: 10.1007/s00158-009-0420-2.

Shang, Wenling et al. (2016). “Understanding and improving convolutional neural networks via concatenated rectified linear units”. In: *International Conference on Machine Learning (ICML) 2016*, pp. 2217–2225. URL: https://arxiv.org/abs/1603.05201.

Smelik, Ruben M. et al. (Jan. 2014). “A Survey on Procedural Modelling for Virtual Worlds”. In: *Computer Graphics Forum* 33.6, pp. 31–50. DOI: 10.1111/cgf.12276.

Sola, J. and J. Sevilla (June 1997). “Importance of input data normalization for the application of neural networks to complex industrial problems”. In: 44.3, pp. 1464–1468. DOI: 10.1109/23.589532.

Srivastava, Nitish et al. (2014). “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”. In: *Journal of Machine Learning Research* 15.56, pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

Sudret, Bruno, Stefano Marelli, and Joe Wiart (2017). “Surrogate models for uncertainty quantification: An overview”. In: *2017 11th European conference on antennas and propagation (EUCAP)*, pp. 793–797. DOI: 10.23919/EuCAP.2017.7928679.

Sun, Luning et al. (2020). “Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data”. In: *Computer Methods in Applied Mechanics and Engineering* 361, p. 112732. URL: https://arxiv.org/abs/1906.02382.

Tan, Heqing et al. (Apr. 2019). “The Combined Distribution and Assignment Model: A New Solution Algorithm and Its Applications in Travel Demand Forecasting for Modern Urban Transportation”. In: *Sustainability* 11.7, p. 2167. DOI: 10.3390/su11072167.

Tripathy, Rohit K. and Ilias Bilionis (Dec. 2018). “Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification”. In: *Journal of Computational Physics* 375, pp. 565–588. DOI: 10.1016/j.jcp.2018.08.036.

Twomey, Janet M and Alice E Smith (1998). “Bias and variance of validation methods for function approximation neural networks under conditions of sparse data”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 28.3, pp. 417–430. DOI: 10.1109/5326.704579.

Van Rossum, Guido and Fred L. Drake (2009). *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace. ISBN: 1441412697.

Vashisht, Raghav (Sept. 2021). *Machine Learning: When to perform a Feature Scaling?* URL: https://www.atoti.io/when-to-perform-a-feature-scaling/.

Vaswani, Ashish et al. (2017). “Attention is all you need”. In: *Advances in neural information processing systems*, pp. 5998–6008. URL: https://arxiv.org/abs/1706.03762.

Veličković, Petar et al. (2018). “Graph Attention Networks”. In: *6th International Conference on Learning Representations (ICLR) 2018*. URL: https://arxiv.org/abs/1710.10903.

Vos, Jonas De (May 2020). “The effect of COVID-19 and subsequent social distancing on travel behavior”. In: *Transportation Research Interdisciplinary Perspectives* 5, p. 100121. DOI: 10.1016/j.trip.2020.100121.

Vrtic, Milenko et al. (2007). “Two-dimensionally constrained disaggregate trip generation, distribution and mode choice model: Theory and application for a Swiss national model”. In: *Transportation Research Part A: Policy and Practice* 41.9, pp. 857–873. DOI: 10.1016/j.tra.2006.10.003.

Wang, Jason, Luis Perez, et al. (2017). “The effectiveness of data augmentation in image classification using deep learning”. In: *Convolutional Neural Networks Vis. Recognit* 11, pp. 1–8. URL: https://arxiv.org/abs/1712.04621.

Wu, Xin et al. (Nov. 2018). “Hierarchical travel demand estimation using multiple data sources: A forward and backward propagation algorithmic framework on a layered computational graph”. In: *Transportation Research Part C: Emerging Technologies* 96, pp. 321–346. DOI: 10.1016/j.trc.2018.09.021.

Xiao, Tianzheng et al. (2021). “Trip Generation Prediction Based on the Convolutional Neural Network-Multidimensional Long-Short Term Memory Neural Network Model at Grid Cell Scale”. In: *IEEE Access* 9, pp. 79051–79059. DOI: 10.1109/access.2021.3083683.

Xiong, Xi et al. (2020). “Dynamic origin–destination matrix prediction with line graph neural networks and kalman filter”. In: *Transportation Research Record* 2674.8, pp. 491–503. DOI: 10.1177/0361198120919399.

Yang, Chao et al. (Nov. 2013). “Sensitivity-based uncertainty analysis of a combined travel demand model”. In: *Transportation Research Part B: Methodological* 57, pp. 225–244. DOI: 10.1016/j.trb.2013.07.006.

Yang, Li and Abdallah Shami (2020). “On hyperparameter optimization of machine learning algorithms: Theory and practice”. In: *Neurocomputing* 415, pp. 295–316. DOI: 10.1016/j.neucom.2020.07.061.

Zannat, Khatun E and Charisma F. Choudhury (Oct. 2019). “Emerging Big Data Sources for Public Transport Planning: A Systematic Review on Current State of Art and Future Research Directions”. In: *Journal of the Indian Institute of Science* 99.4, pp. 601–619. DOI: 10.1007/s41745-019-00125-9.

Zhang, Chao, Carolina Osorio, and Gunnar Flötteröd (Mar. 2017). “Efficient calibration techniques for large-scale traffic simulators”. In: *Transportation Research Part B: Methodological* 97, pp. 214–239. DOI: 10.1016/j.trb.2016.12.005.

Zhang, Qi et al. (Aug. 2018). “Kernel-Weighted Graph Convolutional Network: A Deep Learning Approach for Traffic Forecasting”. In: *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE. DOI: 10.1109/icpr.2018.8545106.

Zhang, Zhilu and Mert R Sabuncu (2018). “Generalized cross entropy loss for training deep neural networks with noisy labels”. In: *32nd Conference on Neural Information Processing Systems (NeurIPS)*. URL: https://arxiv.org/abs/1805.07836.

Zhao, Xilei et al. (July 2020). “Prediction and behavioral analysis of travel mode choice: A comparison of machine learning and logit models”. In: *Travel Behaviour and Society* 20, pp. 22–35. DOI: 10.1016/j.tbs.2020.02.003.

Zhou, Bo et al. (Apr. 2020). “A reinforcement learning scheme for the equilibrium of the in-vehicle route choice problem based on congestion game”. In: *Applied Mathematics and Computation* 371, p. 124895. DOI: 10.1016/j.amc.2019.124895.

Zhou, Zhong, Anthony Chen, and Sze Chun Wong (2009). “Alternative formulations of a combined trip generation, trip distribution, modal split, and trip assignment model”. In: *European Journal of Operational Research* 198.1, pp. 129–138. DOI: 10.1016/j.ejor.2008.07.041.

Zito, Pietro et al. (Jan. 2011). “The effect of Advanced Traveller Information Systems on public transport demand and its uncertainty”. In: *Transportmetrica* 7.1, pp. 31–43. DOI: 10.1080/18128600903244727.