Towards Data-Efficient Mobility Analytics in Spatial Networks

Skovgaard Jepsen, Tobias

DOI (link to publication from Publisher):
10.54337/aau456351766

Publication date:
2021

Document Version
Publisher's PDF, also known as Version of record

TOWARDS DATA-EFFICIENT MOBILITY ANALYTICS IN SPATIAL NETWORKS

BY
TOBIAS SKOVGAARD JEPSEN

DISSERTATION SUBMITTED 2021

AALBORG UNIVERSITY
DENMARK

Towards Data-Efficient Mobility Analytics in Spatial Networks

Ph.D. Dissertation

Tobias Skovgaard Jepsen

Dissertation submitted August 2021

Dissertation submitted: August 2021

PhD supervisor: Prof. Christian S. Jensen
Aalborg University

PhD Co-Supervisor: Assoc. Prof. Thomas Dyhre Nielsen
Aalborg University

PhD committee: Associate Professor Álvaro Torralba (chairman)
Aalborg University

Professor Cyrus Shahabi
University of Southern California

Associate Professor Eleni I. Vlahogianni
National Technical University of Athens

PhD Series: Technical Faculty of IT and Design, Aalborg University

Department: Department of Computer Science

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-977-0

Published by:
Aalborg University Press
Kroghstræde 3
DK – 9220 Aalborg Ø
Phone: +45 99407140
aauf@forlag.aau.dk
forlag.aau.dk

Printed in Denmark by Rosendahls, 2021

Abstract

Vast amounts of vehicle trajectory data is being collected which has solutions to traditional transportation tasks at a finer granularity than ever before, but also enabled solutions to entirely new tasks. However, although vehicle trajectory data presents great opportunity, utilizing such data presents significant challenges.

First, trajectory data is sparse in contextual information. One way to address this issue is to map-match trajectories s.t. they are enriched with map data that provides road segment attribute information and structural information about the road network. Unfortunately, map data contains little attribute information and the structure of the road network is difficult to leverage. Second, vehicle trajectory data is skewed s.t. it concentrates on some road segments at some times of day. This leads to the curious situation where data abundance and data sparsity co-occur in the same road network. Techniques that utilize vehicle trajectory data should therefore be sufficiently flexible to handle both situations. Finally, large sets of vehicle trajectories present challenges in terms of storage and transmission costs. It is therefore beneficial to be capable of processing data in compressed or easy-to-compress formats.

Efficient data utilization is the key to solve transportation tasks using large sets of vehicle trajectory data. In particular, data efficient techniques should utilize the available data as much as possible, and perform well under both conditions of data sparsity and data abundance. Finally, they should also be capable of operating on compressed or easy-to-compress trajectory data formats, and, as a result, the thesis focuses on the use of map-matched vehicle trajectory data, i.e., trajectories represented as road segments in a road network, which can be compressed efficiently. This thesis focuses on data efficient techniques for solving transportation tasks using large sets of vehicle trajectory data.

First, the thesis investigates the application of network representation learning techniques to road networks. These techniques extract information from the structure of the road network, without relying on any attribute information, and can therefore improve the utilization of map data for contextual information when using map-matched vehicle trajectories. However, existing network representation learning techniques make assumptions that are inappropriate for road networks. This thesis therefore presents a representation learning technique designed for road networks that outperforms existing techniques on two transportation tasks.

Second, the thesis investigates how two categories of approaches to the important task travel time and speed estimation can be combined to leverage the strengths of both. The first approach relies on function-fitting and performs well under conditions of data sparsity. The other approach relies on aggregation of historical data and performs well under conditions of data abundance. However, since data abundance and data sparsity co-occur in road networks, no technique is universally applicable to all areas of the road network. This thesis therefore proposes a travel time and speed estimation framework that uses Bayesian probability theory to determine when to apply each approach. Rather than a hard decision rule, the framework smoothly transitions from a function-fitting approach to an aggregation-based approach as the available data increases.

Third, the thesis proposes techniques for analyzing driver behavior based on vehicle trajectories that have been converted into an easy-to-compress format. Specifically, the techniques can discover intermediate destinations within a vehicle trajectory and recover the preferences a driver exhibits in a trajectory w.r.t., e.g., travel time and fuel consumption. The techniques are not only able to operate on easily-compressed vehicle trajectory data, but also has low processing costs with trivial parallelizability s.t. even very large trajectory sets can be processed efficiently.

Resumé

Store mængder turdata fra køretøjer indsamles. Det har tilladt at dels at kunne løse traditionelle transportopgaver med en finere granularitet end hidtil, men også at helt nye transportopgaver nu kan løses. Men selvom de store mængder turdata giver en masse muligheder, så er der også væsentlige udfordringer forbundet i at udnytte dem.

For det første, så indeholder turdata meget lidt kontekstuel information. Èn måde at håndtere dette problem på er ved at berige turdataen med kortdata der indeholder attributinformation om vejsegmenter samt information om vejnetværkets struktur. Desværre er der meget lidt attributinformation i kortdata og det er svært at udnytte informationen om vejnettets struktur. For det andet, så er turdata ulige fordelt over vejnettet således at det koncentrer sig omkring nogle vejsegmenter på nogle tidspunkter. Det leder til en spøjs situation, hvor der kan være både overflod og mangel på data i det samme vejnet. Teknikker til at udnytte turdata bliver derfor nød til at kunne håndtere begge betingelser. Endelig, så er omkostningerne forbundet med opbevaring og transports af store mængder turdata betragtelige. Det er derfor en fordel at kunne processere turdata i komprimerede eller komprimeringsvenlige formater.

Effektiv udnyttelse af turdata er nøglen til at løse transportopgaver ved hjælp af store mængder turdata. Specifikt, så skal dataeffektive metoder kunne udnytte den tilgængelige data i videst mulige omfang, samt kunne fungere både med mangel på data og med overflod af data. Desuden så skal dataeffektive metoder kunne benytte turdata i komprimerede eller komprimeringsvenlige formater. Denne afhandling fokuserer på dataeffektive teknikker til at løse transportopgaver ved brug af store mængder turdata.

Først undersøges det hvorvidt eksisterende netværksrepræsentationslæringsteknikker kan anvendes på vejnet. Disse teknikker kan lave vektorrepræsentationer der indeholder information om et vejnets struktur uden brug af attributinformation. Vektorrepræsentationerne kan supplementere eksisterende attributinformation og således afhælpe problemet med den lave mængde attributinformation der findes i vejnet. Desværre viser det sig at disse repræsentationslæringsteknikker lave antagelser der ikke er rimelige for vejnet og

derfor bidrager denne afhandling med en netværksrepræsentationslæringsteknik der er designet specifikt til vejnet.

Afhandlingen bidrager også med en teknik til estimering og forudsigelse af kørselstider og -hastigheder der kombinerer to eksisterende tilgange til opgaven. Den første tilgang finder en funktion vha. regression og klarer sig godt i situationer med lav datatilgængelighed. Den anden tilgang aggregerer historiske data og klarer sig godt i situationer med høj datatilgængelighed. Hverken den funktionsbaserede tilgang eller den aggregeringsbaserede tilgange tager højde for at der i det samme vejnet vil være både områder med lav og høj datatilgængelighed og at den ideele tilgang vil variere derefter. Derfor præsenterer denne afhandling en hybrid tilgang i form af et rammeværk for forudsigelse af kørselstider og -hastigheder der vha. Bayesiansk sandsynlighedsteori afgør hvornår hvilken tilgang skal anvendes. Det betyder at tilgangen gradvist vil gå fra en funktionsbaseret tilgang til en aggregeringsbaseret tilgang i takt med at mere data bliver tilgængelig.

Endelig, så bidrager afhandlingen med teknikker til at analysere kørselsadfærd baseret på turdata der er lagret i et komprimeringsvenligt format. Teknikkerne kan finde viapunkter i ture, samt førerens kørselspræferencer ift. eksempelvis kørselstid og brændstofforbrug. Udover at kunne anvendes på komprimeret turdata, så kræver teknikkerne lav processeringstid og er trivielt paralleliserbare. De kan således skalere til selv meget store mængder turdata.

Contents

II Papers 29

Appendices **95**

D UniTE—The Best of Both Worlds: Unifying Function-Fitting and Aggregation-Based Approaches to Travel Time and Travel Speed Estimation **101**

Acknowledgments

A PhD-project is no easy endeavor and one that I have found cannot be done alone.

I have had the great fortune of having two supervisors, Christian S. Jensen and Thomas Dyhre Nielsen, who I admire and who have taught me how to think both more broadly and more deeply about my research. I have no doubt not always been easy to supervise, but know that I have always appreciated your time and effort.

For my external stay, I visited the Institute for Formal Methods in Computer Science at the University of Stuttgart. I would like to extend my gratitude towards the faculty of the institute for making my visit an enjoyable experience. I would like to pay special thanks to Stefan Funke, Florian Barth, and Claudius Proissl who I worked with closely during my visit.

To my current and past colleagues at Cassiopeia, but the DAISY group in particular, I would like to thank you all for creating a welcoming and open work environment. I have also had the pleasure of getting to know many of you outside of work. I would also like to thank the administrative personnel who have always been kind and patient when I required their assistance.

I doubt that I could have completed my studies and thesis without the support of friends and family. I thank you all for providing me with a space of safety and comfort whenever I needed it. My dear friend, Martin, has been particularly supportive and especially so during times of personal crisis. So, thank you, Martin, for your unwavering loyalty, for your endless understanding, and for your insatiable curiosity which continues to inspire me.

Tobias Skovgaard Jepsen

August 4, 2021

Thesis Details

Thesis Title:	Towards Data-Efficient Mobility Analytics in Spatial Networks
PhD Student:	Tobias Skovgaard Jepsen
PhD Supervisor:	Prof. Christian S. Jensen Aalborg University
PhD Co-Supervisor:	Assoc. Prof. Thomas Dyhre Nielsen Aalborg University

This thesis is in the format of a collection of papers and consists of the following papers:

A Tobias Skovgaard Jepsen, Christian S. Jensen, Thomas Dyhre Nielsen, and Kristian Torp, 'On Network Embedding for Machine Learning on Road Networks: A Case Study on the Danish Road Network.' *In Proceedings of the 2018 IEEE International Conference on Big Data*, 2018, pp. 3422–3431. Received the Best Paper Award at the 3rd IEEE International Workshop on Big Spatial Data.

B Tobias Skovgaard Jepsen, Christian S. Jensen, and Thomas Dyhre Nielsen. 'Graph Convolutional Networks for Road Networks.' *In Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2019, pp. 460–463.

C Tobias Skovgaard Jepsen, Christian S. Jensen, and Thomas Dyhre Nielsen, 'Relational Fusion Networks: Graph Convolutional Networks for Road Networks.' *IEEE Transactions on Intelligent Transportation Systems*, 2020. In early online access.

D Tobias Skovgaard Jepsen, Christian S. Jensen, and Thomas Dyhre Nielsen, 'UniTE—The Best of Both Worlds: Unifying Function-Fitting and Aggregation-Based Approaches to Travel Time and Travel Speed Estimation.' Submitted to *ACM Transactions on Spatial Algorithms and Systems*.

E Florian Barth, Stefan Funke, Tobias Skovgaard Jepsen, and Claudius Proissl, 'Scalable unsupervised multi-criteria trajectory segmentation and driving preference mining.' *In Proceedings of the 9th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, 2020, Article No. 6, pp. 1–10.

Part I

Thesis Summary

1 Introduction

The use of mobile devices is widespread and many such devices have built-in tracking capabilities through, e.g., Global Positioning System (GPS). This has enabled the collection of large amounts of trajectory data from vehicles [1–3] where each trajectory record the movements of a vehicle over a period of time.

Advances in processing power has enabled the processing of large trajectory data sets using, e.g., deep learning methods [4, 5]. This has lead to finer-granularity solutions to traditional Intelligent Transportation System (ITS) tasks, but also solutions to new ITS tasks. For instance, travel time estimation for routes has traditionally been computed as a sum of independent travel time estimates for each road segment in the route [6]. This ignores that there are often dependencies between the travel times of adjacent road segments due to, e.g., turns or traffic lights, and therefore a paradigm that focuses on the travel time estimation of routes directly has emerged [6]. Other uses of vehicle trajectory data include forecasting traffic conditions [7] and analyzing driver behavior [8].

1.1 Challenges

Although large vehicle trajectory sets present opportunity, they also present challenge.

First, vehicle trajectories are available in large quantities, but they are sparse in contextual information such as speed limits. As a result, vehicle trajectories are map-matched to road networks s.t. they are represented as sequences of road segments rather than sequences of coordinates. Map-matching improves the quality of the trajectories by enriching them with map data which includes road network structure and road segment attribute information. Unfortunately, such attribute information is sparse and the road network structure is difficult to leverage [9].

Second, vehicle trajectory data is skewed s.t. trajectory data tends to be concentrated on certain road segments at certain times of day [1–3]. For instance, vehicle trajectory data may concentrate on arterial roads during rush hours. This leads to a situation where conditions of data abundance and data sparsity co-occur within a road network. ITSs that wish to fully leverage such data must therefore be capable of handling both conditions.

Third, large vehicle trajectory sets incur large storage and transmission costs [10, 11]. These costs can be reduced by using compression techniques [10, 11]. Techniques for map-matched trajectories are particularly efficient, since map-matched trajectories require less storage in the first place and they can utilize the road network to achieve a better compression ratio [12]. This thesis therefore focuses on leveraging map-matched vehicle trajectories for solving ITS tasks.

1.2 Contributions

Efficient data utilization is central to the performance of ITSs that seek to take advantage of large trajectory sets. There are four important aspects to data efficiency. A data efficient technique should

1. utilize the available data to the largest possible extent.
2. perform well when little data is available.
3. be capable of leveraging large amounts of data when it is available.
4. operate on compressed or easy-to-compress data formats to reduce storage and transmission costs.

This thesis addresses three primary research problems related to data efficiency in ITSs.

Road Network Representation Learning

Vehicle trajectories lack contextual data, but through map-matching additional information can be integrated from map data. However, map data is scarce in attribute information and the latent information in the road network structure is difficult to access [9]. It is therefore important to fully utilize the available information.

Representation learning techniques for general networks can encode the difficult-to-access road network structure as road segment vector representations [9, 13, 14]. However, these techniques are not designed for road networks and thus their applicability is uncertain [9, 13, 14]. This thesis therefore investigates the applicability of such techniques for road network representation learning using a case study on the Danish road network [9].

Based on the findings of the case study, this thesis proposes a representation learning technique designed specifically for road networks. The technique relaxes the assumptions of network representation learning techniques concerning the relationships among adjacent road segments. An empirical evaluation of the technique on the tasks of vehicle travel speed estimation and speed limit classification showed improvements in the range of 21%–40% over comparable state-of-the-art techniques. [13, 14]

Travel Time and Travel Speed Estimation

Large vehicle trajectory sets are an appealing data source for point or distribution estimation of travel time (or travel speed) for routes or road segments to enable, e.g., routing applications. Existing approaches to travel time and travel speed estimation has diverged into two branches. The first branch aims to achieve high accuracy under conditions of data abundance and the other branch aims to achieve good generalization performance under conditions of data sparsity. However, both branches ignore that data abundance and data sparsity co-occur as a resulting the data skew inherent in vehicle trajectory data. [15]

This thesis proposes a Bayesian framework that integrates approaches from both branches into a hybrid approach. Using Bayesian probability theory, the framework can use whichever approach is most suitable given the available data, This allows the framework to achieve both high accuracy when data is abundant and good generalization performance when data is sparse. [15]

Vehicle Trajectory Analysis

The route of a map-matched vehicle trajectories can be compressed efficiently, but temporal information such as arrival times at each road segment cannot. This thesis explores how vehicle trajectory analysis may be performed on map-matched trajectories, where only the route of the trajectory is known. Specifically, techniques are proposed that can discover interesting points within a trajectory, e.g., intermediate destinations, and can quantify the driving preferences exhibited within a trajectory. The techniques are evaluated using a set of 1.3 million map-matched trajectories. Interestingly, the techniques can recover temporal information, such as stops, by analyzing just the route of a trajectory. [8]

1.3 Organization

The remainder of this summary is organized as follows. Section 2 summarizes Papers A [9], B [13], and C [14] on the topic of road network representation learning. Section 3 summarizes Paper D [15] which presents the travel time and speed estimation framework. Section 4 summarizes Paper E [8] which presents the vehicle trajectory analysis techniques for map-matched vehicle trajectories. Finally, Section 5 summarizes the contributions of the thesis and concludes the thesis summary. No new claims are made in the summary.

2 Road Network Representation Learning

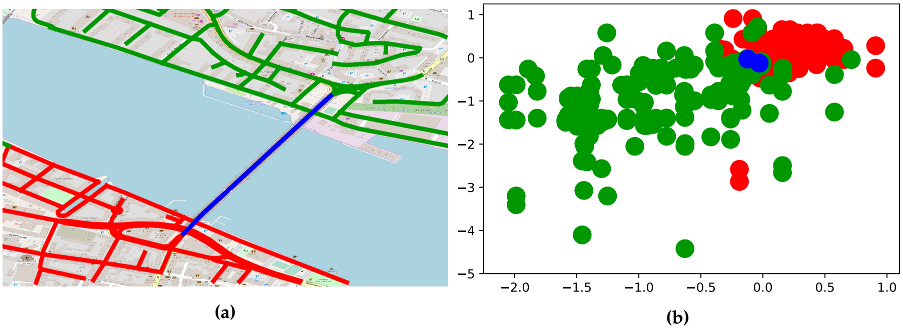
Road networks are an important type of transportation network, relevant to many ITS applications. Such applications often rely on the use of machine learning algorithms that require descriptions of, e.g., road segments in the form of feature vectors, and the performance of these algorithms depends strongly on the quality of the feature vector representations. [9]

In practice, road network data contains little information beyond the road network structure. As an example, the Danish road network from OpenStreetMap (OSM) contains just two road segment attributes: the road category and the speed limit. In addition, only 13% of the road segments have a known speed limit. Constructing feature vectors of sufficient quality from such a small amount attribute data is challenging. The information embedded in the structure of the road network is potentially very valuable, but also hard to access. Utilizing the structural information of a road network typically requires explicit modeling of spatial correlations among adjacent road segments. [9]

Road networks are not the only type of network where the network structure is potentially valuable, but hard to access. Network representation learning techniques aim to learn a mapping from nodes or edges, representing, e.g., intersections and road segments, to a d-dimensional vector space. This way, it becomes possible to map road segments to feature vectors that encode the structural information of a road network. However, the research in network representation methods has focused primarily on social, biological, and information networks. Such networks differ substantially from road networks in terms of structure, semantics, size, etc. This makes their applicability to road networks uncertain. [9]

This thesis explores two kinds of approaches to network representation learning: network embedding methods and graph convolutional networks. Network embedding methods are a class of self-supervised methods that aim to preserve network structure in the embedding space [9]. Thus, road segments that are near in the road network should be near in the embedding space. Network embedding methods are typically used as a data preprocessing step for subsequent machine learning algorithms. Graph Convolutional Networks (GCNs) are supervised methods that are typically integrated into neural networks as representation learning components [13, 14]. In GCNs, road segments inherit the feature vectors of their adjacent road segments.

Both network embedding methods and GCNs rely on the assumption that the networks to which they are applied exhibits homophily, e.g., in road networks, a tendency for similar road segments to be connected [9, 13, 14]. This assumption can be problematic in road networks: residential areas in two different cities may be similar, while being far apart in the road network [9]. In addition, homophily implies smooth transitions in network characteristics, but in road networks transitions are often abrupt [13, 14]. For instance, the travel speed of a vehicle typically changes drastically over a short distance when exiting a motorway [13, 14].



(a)

(b)

Fig. 1: Illustration of (a) road segments in the road network and (b) their two-dimensional vector representations generated by node2vec. Colors indicate the region a road segment belongs to. [9] © 2018 IEEE.

2.1 On the Validity of the Homophily Assumption

This section summarizes the contents of Paper A [9]. Reused content from the paper may appear for the sake of clarity of explanation.

A case study is conducted to assess the validity of the homophily assumption and, by implication,the applicability of state-of-the-art network representation learning methods to road networks because rely on this assumption. Specifically, the network embedding method node2vec is applied to the classification of road categories and speed limits in the Danish road network.

Data

The Danish road network is extracted from OSM and is converted to a directed multigraph $G = (V, E)$. Here, each node $v \in V$ represents an intersection or the end of a road and each edge $e \in E$ represents a road segment. The resulting graph consists of 583 816 nodes and 1 291 168 edges. All road segments belong to one of nine road categories, and 163 043 road segments (ca. 13%) have an associated speed limit.

Node2Vec

The embedding method node2vec (and other network embedding methods) aims to learn a mapping ϕ s.t. similar road segments have similar vector representations in the embedding space. In practice, this is achieved by choosing ϕ s.t. it optimizes the objective

$$\sum_{e \in E} \log \Big(\Pr(N(e) \mid \phi(e)) \Big),$$

where N is a road segment neighborhood function. As illustrated in Figure 1, the result is that road segments that share neighbors are mapped to similar vectors. If the underlying network exhibits homophily, this assumption makes sense: if road segments tend to be connected to similar road segments, then two road segments that share neighbors tend to also be similar.

Homophily in Road Networks

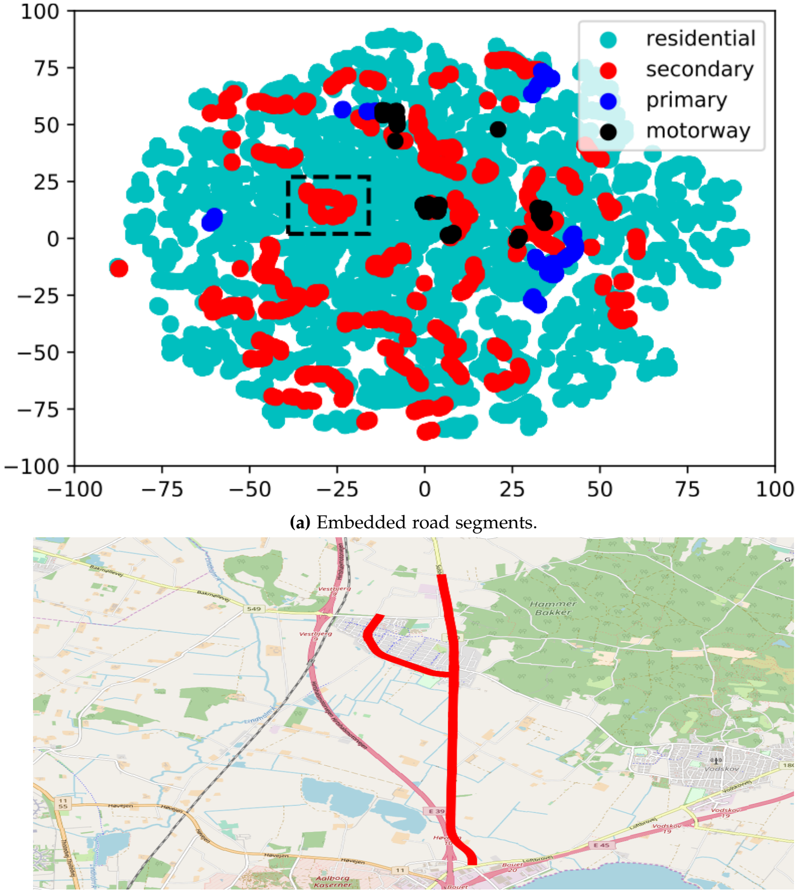
The primary finding of the case study is that road networks exhibit homophily, but the nature of this homophily is differs from the one assumed by network representation learning methods.

In the network embedding literature, linear classifiers are commonly used because their classification performance is a good measure of the quality of the feature vectors produced by a network embedding method. If the performance of the linear classifier is high, this indicates a high degree of linear separability in the embedding space. Linear separability in the embedding space is highly desirable since it simplifies classification tasks.

In the network embedding literature, linear classifiers generally perform quite well. For instance, according to one paper, the macro F1 score of node2vec on node classification on the popular Cora and Citeseer networks is 0.5233–0.5972 and 0.7256–0.8162, respectively [16]. Using the same linear classification algorithm in the case study resulted in scores below 0.2 and 0.3 on road category and speed limit classification, respectively.

The cause of the poor performance of the linear classifier in the case study becomes clear when the distribution of the features vector representations of road segments produced by node2vec is visualized. As shown in Figure 2, the embedding space is far from linearly separable: road segments belonging to the same category are spread across several clusters. As illustrated by the figure, each such cluster corresponds to a group of adjacent road segments. Thus, each cluster of road segments belonging to the same road category is internally homophilic but may be far from other similar clusters in the embedding space. In digital networks, like the ones commonly considered in the network embedding literature, this is less likely to happen since there is no physical limit to the amount of connections associated with a node in the network.

The nature of the distributed homophily in road networks leads to poor performance when using the resulting node2vec embedding in conjunction with the linear classifier used in the case study. However, network embedding methods may still be useful for machine learning on road networks: in the case study, a non-linear random forest classifier achieves promising macro F1 scores of 0.57 and 0.79 for speed limit and road classification, respectively. These scores are in line with typical performance numbers seen in the network embedding literature when linear classifiers are used.



(a) Embedded road segments.

(b) Spatial representations of the framed cluster in Figure 2a.

Fig. 2: Two-dimensional t-distributed Stochastic Neighbor Embedding (t-SNE) projections of the road segment embeddings in Aalborg Municipality. [9] © 2018 IEEE.

Discussion and Conclusion

The case study finds that the homophily found in road networks is different from the homophily found in the networks network embedding methods are designed for. Road segments appear in internally homophilic clusters where similar clusters may be far apart in terms of network distance. As a consequence of the neighborhood-preserving network embedding objective, the clusters are also far apart in the embedding space, resulting in an embedding space with a low degree of linear separability. Compared to applications of network embedding techniques in the literature, these results indicates a substantial degradation of embedding quality when applied to road networks, necessitating the use of powerful non-linear classifiers, such as the random forest classifier used in the case study. Future network embedding methods should therefore emphasize the capture of capture information about the structural similarities among road segments rather than relying on their connectivity.

Generalizability of the Case Study: The case study uses just one network embedding for just two classification tasks in just one road network. Yet, the findings are expected to generalize beyond this case study.

First, the adjacency-preserving objective used by node2vec, where adjacent road segments are near in the embedding space, is used universally in network embedding methods to exploit homophily. The findings are therefore expected to generalize to other network embedding methods. Second, the distribution of road segments into several internally homophilic areas that may be far apart is a natural consequence of countries being divided into areas of different qualities, e.g., cities, and therefore the findings are expected to generalize to other road networks.

Third, the two tasks of speed limit and road category classification are closely linked to the division of road networks into different areas. For instance, speed limits are typically low in residential areas. However, many other important quantities of interest exhibit the same behavior, such as traffic congestion levels and travel speed which also differ among, e.g., urban and rural areas. The findings are therefore expected to be relevant for many other prediction tasks on road networks.

2.2 Graph Convolutional Networks for Road Networks

This section summarizes the contents of Paper B [13], and its extended version, Paper C [14]. Reused content from the papers may appear for the sake of clarity of explanation.

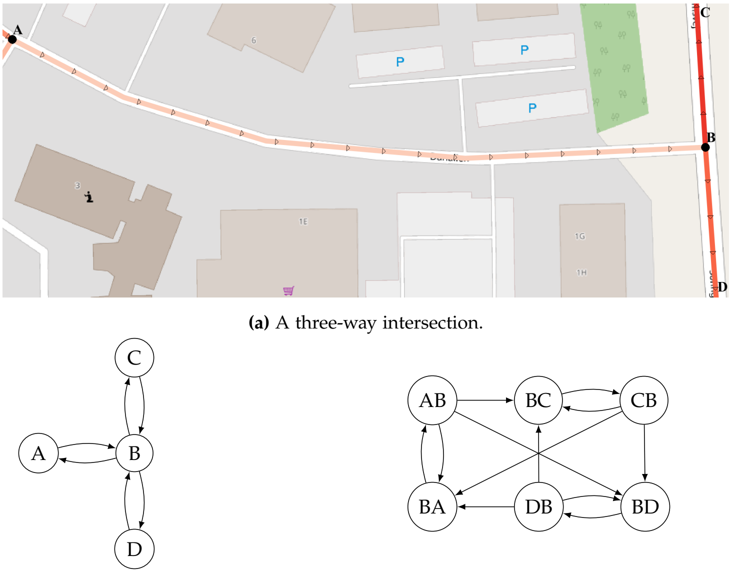
A Graph Convolutional Network (GCN) is a neural network that takes as input a graph representation of a network. As in the case of the network embedding methods discussed in Section 2.1, GCNs aim to encode the structure of a network into feature vector representations of, e.g., road segments. Unfortunately, GCNs have been designed in the context of the same types of digital networks that network embedding methods have and they rely on the same notion of homophily being present in the network. Although road networks are homophilic, it takes a different form than in these kind of networks.

As discussed in Section 2.1, road segments tend to be connected to other similar road segments, but similar road segments belong to clusters that may be far apart in the network. As an example, two internally homophilic residential areas may be similar, but they are located in different cities. In addition, the homophily is volatile in the sense that changes can occur rapidly. For instance, a motorway road segment and an urban road segment may be separated by just one road segment, e.g., a motorway exit. State-of-the-art GCNs have not been designed with these properties in mind. To address the need for representation learning methods for road networks, Relational Fusion Networks (RFNs) are proposed.

Primal and Dual Graph Representations of Road Networks

Traditionally, a road network is modeled as a directed graph $G = (V, E)$, where V is a set of nodes representing intersections or the end of road, and E is a set of edges representing road segments. This graph representation, called the primal graph representation, models the relationships between intersections and can be used to computed representations of intersections. Supplying the primal graph representation to a GCN yields feature vector representations for making predictions over intersections. Typically, however, one is interested in making predictions over road segments based on their relationships to adjacent road segments. To this end, a dual graph representation of the road network may be given to the GCN instead.

A dual graph representation of a road network models the relationships between road segments, rather than intersection, s.t. road segments are represented as nodes and there exists an edge between two road segments if they are connected through an intersection. This is illustrated in Figure 3 using a three-way intersection. The primal and dual graph representations are related. Let $G^P = (V, E)$ denote the primal graph representation of a road network. Then, the dual graph representation of the road network is $G^D = (E, B)$, where B is the set of between-edges. Formally, there exists a between-edge between two edges (u, v) and (w, y) in E if $v = w$. An added advantage of the dual graph representation when used in conjunction with a GCN, is that it naturally yields representations of road segments.



(a) A three-way intersection.

(b) Primal Graph.

(c) Dual Graph.

Fig. 3: (a) A three-way intersection, and its (b) primal and (c) dual graph representations. [13, 14] © 2020 IEEE.

Graph Convolutional Networks

GCNs are similar to network embedding methods in that they try to take advantage of homophily in the underlying network to produce useful representations of nodes or edges. However, where network embedding methods are concerned with the objective that is to be optimized [9], GCNs are concerned with the structure of the function being optimized [13, 14]. Network embedding methods may even utilize a GCN as the embedding function [17].

GCNs are neural networks with architectures designed to utilize the structure of the given graph representation of a network to take advantage of homophily in the network. Formally, GCNs consists of graph convolutional layers s.t. the representation of a node v at the kth layer is

$$\mathbf{H}_v^{(V,k)} = \sigma\big(\textsc{Aggregate}^k(\{\mathbf{X}_n \mid n \in N(v)\})\mathbf{W}^k\big), \tag{1}$$

where σ is an activation function, N is a function that returns they neighbors of node v, $\mathbf{X}_n$ is an input feature vector representation of node n, and $\mathbf{W}^k$ is a weight matrix of learnable model parameters. The function $\textsc{Aggregate}^k$ computes an aggregated representation of the neighborhood using, e.g., the mean representation of v's neighbors.

GCNs takes advantage of homophily through aggregation of neighbor representations. However, a property of the aggregation functions used in the GCNs literature is that they will assign similar vector representations to nodes that have large overlaps in their neighborhood. This property is also reflected in the output of each graph convolutional layer in a GCN and allows the GCN to exploit homophily in the underlying network.

Direct Inheritance and Volatile Homophily

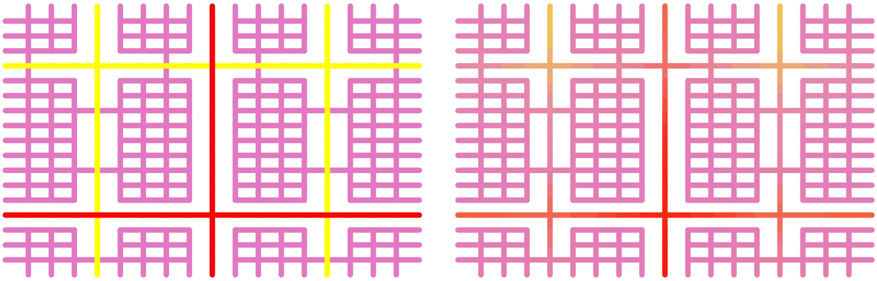
GCNs computes road segment representations based on aggregation of the feature vector representations of adjacent road segments. Thus, two road segments achieve similar representations if they are adjacent to similar road segments, even if they are far apart in the network and share no neighbors. However, when producing representation for a road segment, state-of-the-art GCNs rely on direct inheritance of the feature vector representations of adjacent road segments. This means that two road segments with an overlap among their adjacent road segments are given similar feature representations, even if those road segments are quite different.

Direct inheritance works well in networks that exhibit smooth homophily, where changes in network characteristics happen slowly. In contrast, homophily in road networks is often volatile. For instance, when exiting a motorway into an urban area, the travel speed drop substantially across a short distance.

An Illustrative Example: Figure 4a illustrates an urban area in a road network, where the yellow road segments give access to different residential areas consisting of teal road segments. The red segments are major road segments that connect different this area to the rest of the city. These colors represent network characteristics such as travel speed. As shown in the figure, the homophily is volatile: road segments with different characteristics are adjacent.

To illustrate the problem of direct inheritance in the face of volatile homophily, each road segment is given a feature vector representation corresponding to its three-values decimal RGB color code in Figure 4a. Then, a new color is computed by feeding these feature representations through a GCN, where a new color is computed by taking a simple mean of the colors of the adjacent neighbors. Despite its simplicity, this aggregation mechanism has achieved very good performance on standard datasets used in the GCN literature [17].

As shown in Figure 4b, the aggregation mechanism can recover the original color well when deep within residential areas, where there is a high degree of homophily, but it performs poorly on road segments that connect areas with different network characteristics. The effect of the aggregation is a smoothing effect that is suitable for a network exhibiting smooth homophily, i.e., a network where network characteristics slowly change.



(a) Ground Truth. **(b)** GCN Prediction.

Fig. 4: An illustrative example of a road network (a) with ground truth colors representing network characteristics, and (b) the colors predicted by a GCN when given the full colored road network as input.Figures (a) and (b) originate from a poster presentation of Paper B [13].

Relational Fusion Networks

Relational Fusion Networks (RFNs) are proposed to address the short-comings of state-of-the-art GCNs in the context of road networks. In brief, an RFN is a type of GCN, where a relational fusion operator that produces representations of relations is inserted into the regular neighborhood aggregation mechanism. This enables an RFN to aggregate over representations of relations (represented by edges), rather than neighbors, where the representation of the relation is dependent on the attributes associated with both the source and target of the relation, but also on the attributes associated with the relation itself. Using the relation fusion operator, an RFN is not required to use direct inheritance, but can do so if it is useful.

Relational Fusion: Inserting the relational fusion operator into forward propagation of a regular GCN (cf. Equation 1) yields

$$\mathbf{H}_v^{(V,k)} = \sigma\Big(\textsc{Aggregate}^k\left(\{\textsc{Fuse}^k(\mathbf{X}_v, \mathbf{X}_{(v,n)}, \mathbf{X}_n) \mid n \in N(v)\}\right)\mathbf{W}^k\Big), \tag{2}$$

where function Fuse is the relational fusion operator, and $\mathbf{X}_v$, $\mathbf{X}_n$, and $\mathbf{X}_{(v,n)}$ are feature representations of node v, v's neighbor node n, and the edge (v,n) connecting them, respectively. Many different relational fusion operators may be used, but the thesis proposes two such operators: AdditiveFuse [13, 14] and InteractionalFuse [14]. For brevity, this summary reviews only AdditiveFuse. Information regarding InteractionalFuse may be found in Paper C [14].

The definition of AdditiveFuse is

$$\textsc{AdditiveFuse}^k(\mathbf{X}_v, \mathbf{X}_{(v,n)}, \mathbf{X}_n) = \sigma(\mathbf{X}_v\mathbf{W}_1 + \mathbf{X}_{(v,n)}\mathbf{W}_2 + \mathbf{X}_n\mathbf{W}_3) \tag{3}$$

where matrices $\mathbf{W}_1$, $\mathbf{W}_2$, and $\mathbf{W}_3$ are weight matrices.

It is apparent from Equation 3 that inserting AdditiveFuse in Equation 2 yields a generalized version of the GCN forward propagation in Equation 1. Specifically, the regular GCN forward propagation is occurs when $\mathbf{W}_1$ and $\mathbf{W}_2$ are zero matrices, and $\mathbf{W}_3$ is the identity matrix. It follows that AdditiveFuse may use direct inheritance during regular GCN aggregation if it is useful, but may disregard it if it is not.

Other RFN Features: Besides the relational fusion operator addressing the problem of direct inheritance in the face of volatile homophily, RFNs have additional features that makes them particularly suitable for machine learning on road networks.

First, RFNs can incorporate three sources of attribute information—node, edge, and between-edge attributes—by operating on both the primal and dual graph representations in parallel. Examples of node, edge, and between-edge attributes are, respectively, the presence of a traffic light at an intersection, the speed limit of a road segment, and the turn angle between two adjacent road segments. In comparison, regular GCNs support only node attributes, although they can be extended to support edge attributes with relative ease.

Second, RFNs feature an attention mechanism at each layer that allows them to regulate the contribution of each relation to the aggregate in Equation 2. This is particularly important in road networks, where the number of relations an intersection or road segment participates in is typically quite low, so that even a single aberrant or uninformative relation can have a large effect on the aggregate.

An empirical study finds that, in combination, the features of the RFN result in performance increases of $21 - -40\%$ over state-of-the-art GCN architectures on two road network machine learning tasks.

2.3 Conclusion

The thesis investigates the applicability of network representation learning methods for road networks.

Paper A [9] investigates applicability of network embedding methods using a case study. The network embedding method node2vec was applied to the Danish road network, and the resulting feature vector representations of road segments were used to classify road categories and speed limits.

Network embedding methods such as node2vec rely on a strong notion of homophily, where adjacent road segments are expected to be similar. The case study therefore focuses on investigation of the validity of the homophily assumption in road networks. It finds that, while homophily is present in road networks, it is of a different nature than the homophily assumed in network embedding methods. In particular, similar road segments in, e.g., residential areas, may be far apart in the road network as natural consequence of road networks being physical networks. [9]

Papers B [13] and C [14] investigates another class of network representation methods known as GCNs. Like network embedding methods, GCNs also rely on the homophily assumption, but are more suitable for the kind of homophily found in road networks. Specifically, GCNs rely on an aggregation mechanism that summarizes the characteristics of adjacent road segments. Thus, GCNs can assign similar feature vector representations to distant road segments given that the characteristics of their adjacent road segments are similar.

The homophily assumed in GCNs is less strict than the one assumed in network embedding methods, but it fails to account for the volatile nature of homophily in road networks [9, 13, 14]. For instance, motorway approaches often connect road segments with large differences in travel speed, e.g., an urban segment and a motorway segment [13, 14]. GCNs assume far more gradual changes in network characteristics and thus do not have appropriate mechanisms for the volatile homophily found in road networks [13, 14].

The use of direct inheritance during GCN aggregation is identified as the primary culprit. Specifically, GCNs assumes that a road segment shares characteristics with all of its adjacent road segments. However, often this is not the case. For instance, a motorway approach that connects an urban area and a motorway is quite dissimilar from both of its neighbors. To address this issue, RFNs are proposed, which feature a revised aggregation mechanism that allows an RFN to use direct inheritance when it is suitable and ignore it when it is not. [13, 14]

An empirical study finds that RFNs achieve superior performance compared to state-of-the-art GCNs on two machine learning tasks on road networks. [13, 14]

3 Travel Time and Travel Speed Estimation

This section summarizes the contents of Paper D [15]. Reused content from the paper may appear for the sake of clarity of explanation.

Massive amounts of information about travel times and travel speeds along routes and road segments in a network road network can be collected from vehicles equipped with location-tracking equipment. These large amounts of data has enabled fine-granularity estimation tasks where, e.g., travel times along routes are modeled as time-dependent stochastic variables where the travel times of road segments along the route are dependent on each other. However, despite the massive amounts of data, the data tends to concentrate in certain areas and at certain times of day, with little to no data in other areas or at different times. Thus, data abundance and data sparsity

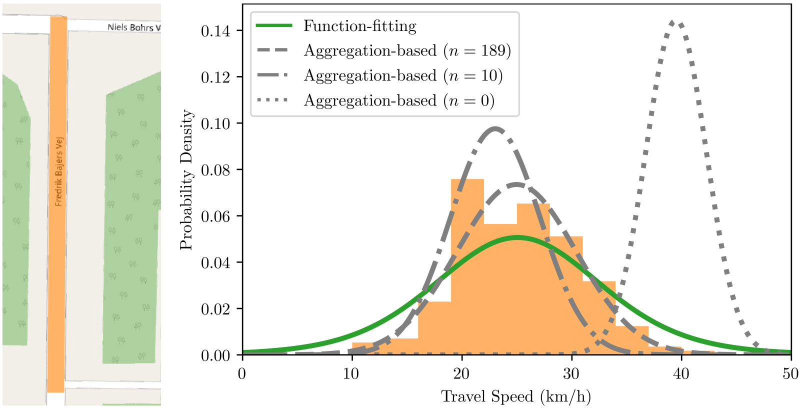


Fig. 5: A road segment (left) and its ground-truth and estimated travel-speed distributions during a time-of-day interval (right) [15].

coexist in a road network, which makes fine-granularity estimation across an entire road network challenging. Two types of approaches to travel time and travel speed estimation exists that target settings with abundant or sparse data.

The first type, aggregation-based approaches, assume that data is generally abundant and that accurate estimates of the travel time (or speed) distribution of a route at a given time can be achieved by aggregating historical traversal times from vehicle trajectories. In the event that no historical travel times are available, the travel speed distribution is found by splitting a route into subroutes s.t. a sufficient number of historical travel times is available for each. If that is not possible, simple-but-inaccurate heuristics based on the speed limit are used, resulting in poor generalizability.

The second type, function-fitting approaches, operate on the assumption that data is generally sparse, i.e., that some or many routes have few or no historical travel times for some or many times of day. They use function-fitting techniques, including deep learning, to map feature representations of routes to travel time distributions, which allows them to achieve high generalizability. In theory, it is possible to fit a fine-grained function, suggesting that function-fitting approaches can achieve estimation accuracies similar to those of aggregation-based approaches in data-abundant situations. However, the data collection and feature engineering efforts required makes it infeasible to do so. In practice, function-fitting approaches trade accuracy in data-abundant situations for generalizability in data-sparse situations.

In summary, aggregation-based and function-fitting approaches represent an accuracy-generalizability trade-off: aggregation-based approaches favor accuracy, and function-fitting approaches favor generalizability. This trade-off is illustrated in Figure 5. Given the full data with $n = 189$, the aggregation-based approach achieves the best fit, whereas the function-fitting approach overestimates the variance. However, when less data is used by the aggregation approach, its performance declines rapidly, and at $n = 0$, it relies on a heuristic that is highly inaccurate.

Both aggregation-based and function-fitting approaches ignore the typical situation, where some vehicle trajectories are highly abundant in some areas at some times while other areas have little or no data associated with them at any time. For instance, in cities, data is typically abundant on arterial roads during rush hours. Often, the available data in an area is directly proportional to the population density. Given this situation, there is a need for flexible approaches that can leverage the accuracy of aggregation-based approaches in data-abundant situations and the generalizability of function-fitting approaches in data-sparse situations. To this end, Paper D proposes a Unifying approach to Travel time and speed Estimation (UniTE).

3.1 UniTE

UniTE is a framework for travel time and travel speed estimation that aims to unify aggregation-based and function-fitting approaches. In brief, the framework uses Bayesian probability theory to gradually transition from a function-fitting approach to an aggregation-based approach as the available data increases. The framework can integrate existing function-fitting and aggregation-based approaches and is thus complementary to both types of approaches The transition between approaches is gradual and mediated by Bayesian probability theory.

Conceptual Model

Figure 6 shows the conceptual model of UniTE. The travel time or speed distribution of a route $\mathbf{p}_i$ at time $\boldsymbol{\tau}_i$ is assumed to follow a distribution with uncertain hyperparameters θ_i. These uncertain parameters follow a prior distribution $\Pr(\theta_i \mid f(\mathbf{p}_i, \boldsymbol{\tau}_i; \psi))$ parameterized by the output of a *prior function* f with function parameters ψ. The prior function f represents the function-fitting component of UniTE that allows a UniTE to inherit the generalizability of the function-fitting approach used as the component. Thus, a UniTE model can estimate travel time or travel speed distributions even if no historical data is available.

An aggregation-based approach is used to select or construct m historical records $\tilde{T}_i = \{t_{i,1}, \ldots, t_{i,m}\}$ that are inserted as evidence in the model, as illustrated in Figure 6. These records represent past traversals of the route at similar, e.g., times of day, in the past, and record the travel time or travel speed of a past traversal. Inserting these historical records as evidence allows the computation of the posterior distribution $\Pr(\theta_i \mid \tilde{T}_i, f(\mathbf{p}_i, \boldsymbol{\tau}_i; \psi))$ over

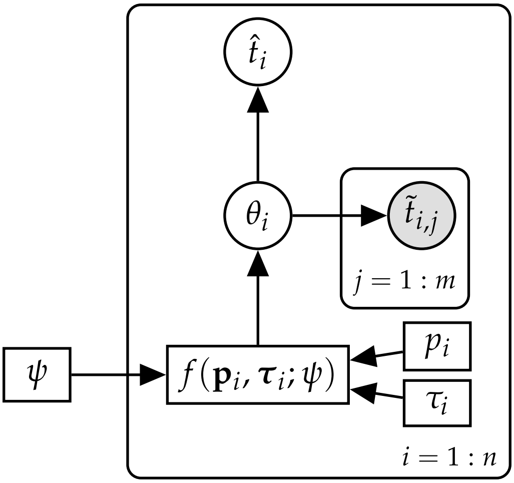


Fig. 6: The UniTE framework illustrated using plate notation. [15]

the uncertain hyperparameters θ_i that involves an aggregation over historical records $\tilde{T}_i$. Thus, the computation of the posterior is representative of an aggregation-based approach in UniTE.

Once a posterior distribution over θ_i has been found, the posterior predictive $\Pr(\hat{t}_i \mid \tilde{T}_i, \theta_i) = \Pr(\hat{t}_i \mid \tilde{T}_i, \mathbf{p}_i, \boldsymbol{\tau}_i; \psi)$ of the travel time or speed $\hat{t}_i$ can be computed and output as the travel distribution of the input route $\mathbf{p}_i$ at time $\boldsymbol{\tau}_i$.

Training a UniTE Model

UniTE models are trained for predictive performance by finding the parameters ψ that maximize the conditional likelihood

$$\Pr(t_i \mid \tilde{T}_i, \theta_i) \tag{4}$$

across n training trajectories, where t_i is the ground truth travel time or travel speed observed in the ith trajectory when traversing route p_i at time τ_i.

During training of a UniTE model, it is recommended to simulate the situation during training where the model relies on the generalizability of its function-fitting component if little or no data is available. This can be achieved by ensuring that $t_i \notin \tilde{T}_i$ in Equation 4, that in turn ensures that the function-fitting component always contributes information about the ground-truth travel time t_i that is missing from the set of historical records $\tilde{T}_i$.

Properties of UniTE

The use of Bayesian probability theory to unify the function-fitting and aggregation-based components in a UniTE model leads to two interesting properties.

First, there always exists a UniTE model that can achieve the same estimation performance as either its function-fitting or aggregation-based component with arbitrary precision, which is particularly interesting when integrating two existing approaches. The higher the confidence in the prior distribution of θ_i, the more the estimation performance of the UniTE model approaches the estimation performance of its function-fitting component. Conversely, the lower the confidence in the prior distribution of θ_i, the more the estimation performance of the UniTE model approaches the estimation performance of its aggregation-based component.

Second, optimizing the conditional likelihood in Equation 4 has a regularizing effect on the function-fitting component s.t. it t performs particularly well in data-sparse situations. This follows from the definition of the posterior predictive used in Equation 4, i.e.,

$$\Pr(\hat{t}_i \mid \tilde{T}_i, \theta_i) = \int_{\theta_i} \Pr(\hat{t}_i \mid \theta_i) \Pr(\theta_i \mid \tilde{T}_i) d\theta_i,$$

which depends on the posterior distribution

$$\Pr(\theta_i \mid \tilde{T}_i) \propto \Pr(\theta_i) \prod_{j=1}^{m} \Pr(\tilde{t}_{i,j} \mid \theta_i). \tag{5}$$

The number of factors in the product in Equation 5 increases as the number of historical records increases s.t. the prior $\Pr(\theta_i)$ has only little influence on the final product when m is large. Thus, the importance of the prior distribution $\Pr(\theta_i)$ on the posterior distribution, and consequently the posterior predictive, is inversely proportional to the number of historical records. Hence, the influence of the function-fitting component on the UniTE objective in Equation 4 is largest when there are no historical records at all, and it gradually becomes less important as more historical records become available. As a result, the function-fitting component is trained to perform well in data-sparse situations when maximizing the conditional likelihood in Equation 4.

3.2 Conclusion

The thesis proposes UniTE, a Bayesian travel time and travel speed estimation framework that integrates existing approaches to travel time and speed estimation into a cohesive framework. Specifically, UniTE models can leverage the generalizability of function-fitting approaches in data-sparse situations and the accuracy of aggregation-based approaches in data-abundant situations.

An empirical study using a simple instance of UniTE, where the travel time is assumed to follow a Gaussian distribution with uncertain mean and variance, finds that UniTE can achieve 40–64% and 3–23% better performance in terms of travel speed distribution modeling and travel time point estimation, respectively, compared to using a function-fitting or aggregation-based approach alone.

4 Vehicle Trajectory Analysis

This section summarizes the contents of Paper E [8]. Reused content from the paper may appear for the sake of clarity of explanation.

Given a trajectory from S to T, drivers are often assumed to choose routes that are optimal in the sense of being, e.g., the fastest or shortest route from S to T. However, in practice, drivers often deviate from such optimal routes due to intermediate destinations such as stopping for gas or shopping on the way home. As an example, consider the trajectory in Figure 7, which goes from S to T with two stops, labeled B. These stops indicate that the driver took a break. Drivers often deviate from optimal routes due to changing intentions or destinations while driving, or due to individual preferences w.r.t. to combinations of costs.

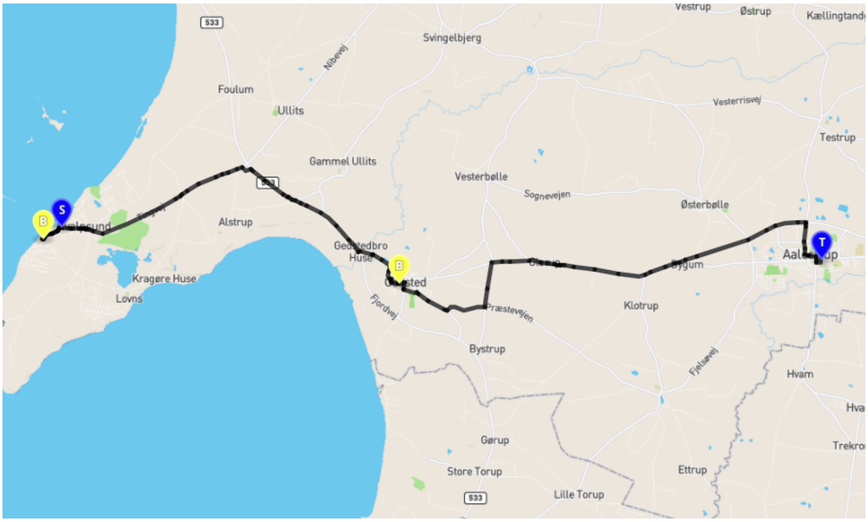


Fig. 7: An example of a trajectory going from S to T with two intermediate stops that are labeled B. [8]

Paper E [8] proposes vehicle trajectory analysis techniques to explain driving behavior in a trajectory. Specifically, the paper proposes a technique for identifying via-points along a trajectory is proposed and a technique for driving preference mining.

4.1 Personalized Costs

Drivers that choose different routes do so because they have different perceived costs or value w.r.t. to some criteria. To model this behavior, personalized traversal costs are introduced in the following.

In the case where a route consists of a single road segment, the following definition of personalized cost is used.

Definition 4.1. *The personalized cost of a road segment e is $p(e \mid \boldsymbol{\alpha}) = \mathbf{c}_e \cdot \boldsymbol{\alpha}$ where $\mathbf{c}_e = (c_{e,1}, \ldots, c_{e,d})$ is a d-dimensional* cost vector *containing d different costs associated with traversal of segment e, and $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_d)$ is a d-dimensional* preference vector, *representing the driving preferences of a particular driver, where $\alpha_i \geq 0$ and $\sum \alpha_i = 1$.*

Using the definition of a personalized cost of a road segment, the personalized cost of a route is defined as follows.

Definition 4.2. *The personalized cost of a route $\pi = (e_1, e_2, \ldots, e_k)$ in a road network is $p(\pi \mid \boldsymbol{\alpha}) = \sum_{i=1}^{k} p(e_i \mid \boldsymbol{\alpha})$.*

4.2 Via-Point Identification

The proposed method for identifying via-points along a trajectory first segments a trajectory into subtrajectories and uses these subtrajectories to identify via-points. For instance, let $T = (v_0, \ldots, v_i, \ldots, v_n)$ be a trajectory that has been segmented into two subtrajectories $(v_0, \ldots, v_i)$ and $(v_i, \ldots, v_n)$. In this example, node v_i represents a via-point in the original trajectory.

Efficient algorithms for segmenting trajectories based on some criterion already exists. The primary contribution of the proposed method is the criterion itself. In brief, the via-point identification algorithm assumes that drivers follow paths that are optimal w.r.t. to their preferences and that any deviations from the optimal path indicates a via-point.

Trajectory Segmentation

For trajectory segmentation, an existing greedy trajectory segmentation algorithm is used. For clarity of the explanation, a simple incremental version of this algorithm is included in Algorithm 1.

Algorithm 1 takes as input a trajectory $T = (v_0, \ldots v_n)$ and a test function Test. The Test function determines whether an input trajectory fulfills some

Algorithm 1 Trajectory Segmentation Algorithm by Buchin et al. [18]

```
 1: function TrajectorySegmentation($T$, Test)
 2:     $\mathcal{T} \leftarrow \emptyset$
 3:     $s \leftarrow 0$
 4:     repeat
 5:         for $i \leftarrow s + 1$ to $n$ do
 6:             $T_i \leftarrow (v_s, \dots, v_i)$
 7:             if not Test$(T_i)$ then
 8:                 $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_{i-1}\}$
 9:                 $s \leftarrow i - 1$
10:                 break
11:     until $i = n$
12:     $\mathcal{T} \leftarrow \mathcal{T} \cup \{T_n\}$
13:     return $\mathcal{T}$
```

criterion, e.g., whether it follows the shortest path. In lines 4–11, the input trajectory is scanned in the **repeat-until** loop and divided into subtrajectories in the **for** loop in lines 5–10. On the first iteration of the for loop, the subtrajectory (v_0, v_1) is tested and is incrementally extended in each iteration of the **for** loop. If the subtrajectory fails the test in line 7, the last subtrajectory T_{i-1} that passed the test is added to the test of subtrajectories $\mathcal{T}$ in line 8. Then, the algorithm exits the **for** loop in line 10, and repeats the procedure starting with subtrajectory (v_{s-1}, v_s). Eventually, the last subtrajectory is extended to the full remaining length of the input trajectory T without failing the test in line 7. At this point, the **until** condition in line 11 is fulfilled, and the **repeat-until** loop exits. The final trajectory T_n is then added to the set of subtrajectories $\mathcal{T}$ in line 12, before $\mathcal{T}$ is returned in line 13.

The Personalized Path Criterion

The subtrajectories produced by Algorithm 1 depends heavily on the criterion in the Test function. The primary contribution of the proposed via-point identification method is the *personalized path criterion*.

Definition 4.3. *A personalized path $\pi_{\boldsymbol{\alpha}} = (e_1, \dots, e_n)$ from s to t for a driver with preference vector $\boldsymbol{\alpha}$ is the path from s to t that has the smallest the personalized cost $p(\pi \mid \boldsymbol{\alpha})$.*

When using the personalized path criterion in Algorithm 1, the Test returns true if an input trajectory T from s to t follows any personalized path from s to t. In other words, a subtrajectory passes the test if some preference vector α, s.t. the subtrajectory is identical to $\pi_{\boldsymbol{\alpha}}$. In practice, determining whether such a preference vector exists can be done by solving the following linear program for which efficient algorithms already exists.

$$\begin{aligned}
\textsc{Minimize} \quad & 1 \\
\textsc{subject to} \quad & \forall \pi \in \Pi \colon p(T \mid \alpha) - p(\pi \mid \alpha) \leq 0 \\
& \sum_{i=1}^{d} \alpha_i = 1 \\
& \forall i \in \{1, \ldots, d\} \colon \alpha_i \geq 0
\end{aligned} \tag{6}$$

Here, Π is the set of all paths from the source vertex s to the target vertex t of trajectory T. Using the personalized path criterion, $\textsc{Test}(T)$ returns true if a solution to the linear program exists, and it returns false if no solution exits.

Using the personalized path criterion in Algorithm 1 yields a trajectory segmentation based on deviations from the path that is optimal w.r.t. to the personalized cost of going from s to t. These points of deviation are marked as via-points.

4.3 Driving Preference Mining

The linear program in Equation 6 implicitly recovers a driving preference vector, but it only has a solution if a driving preference vector $\boldsymbol{\alpha}$ exists s.t. T is a personalized path given $\boldsymbol{\alpha}$. In practice, this makes the linear program rather fragile in terms of driving preference mining since such a preference vector often does not exist

$$\begin{aligned}
\textsc{Minimize} \quad & \delta \\
\textsc{subject to} \quad & \forall \pi \in \Pi \colon p(T \mid \alpha) - p(\pi \mid a) \leq \delta \\
& \sum_{i=1}^{d} \alpha_i = 1 \\
& \forall i \in \{1, \ldots, d\} \colon \alpha_i \geq 0 \\
& \delta \geq 0
\end{aligned} \tag{7}$$

To increase the robustness of the linear program in Equation 6 a few minor modifications are introduced, as shown in Equation 7. First, the solution to Equation 6 is always a preference vector that for the entire trajectory rather than for a subtrajectory. Second, δ is introduced into the first constraint s.t. trajectory T is not required to follow any personalized path. Third, δ is minimized s.t. the solution to the linear program is the driving preference vector $\boldsymbol{\alpha}$ that minimizes the discrepancy between trajectory T and the personalized path w.r.t. $\boldsymbol{\alpha}$. These modifications ensure that a preference vector $\boldsymbol{\alpha}$ always exists as a solution to Equation 7 and that this solution is the preference vector that best explains the driving behavior in the trajectory.

4.4 Conclusion

This thesis proposes methods for trajectory analysis that can identify via-points along a trajectory and mine of driving preferences from a trajectory. Unlike previous work, the techniques utilize just the structure of the trajectory without relying on any additional information such as timestamps.

Despite not utilizing any temporal information, an empirical evaluation shows that the via-point identification technique can recover such information to some extent. The driving preference mining technique can recover driving preference vectors that, when used for personalized routing, can produce routes with 74% overlap with the route of the original trajectory and with a cost similarity of 87%. Finally, both techniques can be implemented efficiently to scale to datasets of millions of trajectories.

5 Conclusion

Vehicle trajectory data presents both opportunity and challenge. The massive volumes of vehicle trajectory data has enabled finer-granularity solutions to traditional Intelligent Transportation System (ITS) tasks as well as solutions to entirely new tasks. However, vehicle trajectories are sparse in contextual information. This problem can be alleviated to some degree by enriching the trajectories with map data, but the map data itself is also sparse. In addition, vehicle trajectory data is skewed s.t. conditions of data abundance and data sparsity co-occur in the same road network. Finally, storage and transmission costs of large vehicle trajectory sets are prohibitive.

To address the challenges associated with the use large vehicle trajectory sets, there is a need for data efficiency in the ITSs that aim to utilize such data. In particular, such systems should utilize the available data as much as possible, and perform well under both conditions of data sparsity and data abundance. Finally, they should also be capable of operating on compressed or easy-to-compress trajectory data formats, and, as a result, the thesis focuses on the use of map-matched vehicle trajectory data, i.e., trajectories represented as road segments in a road network, which can be compressed efficiently. With data efficiency in mind, this thesis has pursued three primary paths of inquiry.

First, the map data used to enrich vehicle trajectories contains little attribute information and the road network structure contained in the map data is difficult to leverage. The problem of how to exploit network structure is not unique to road networks, and network representation learning methods for general networks exists. To investigate the applicability of such methods in the context of road networks, a case study is conducted. The case study finds that while applicable to road networks, the methods makes assumptions that are inappropriate for road networks. The findings of the case study lead to the development of the Relational Fusion Network (RFN), a graph representation learning method for road networks that can be used as a building block in a graph neural network. RFNs relaxes the assumptions of existing graph representation learning methods for general networks, but cannot explicitly leverage the spatiality of a road network or model the temporal variations in the traffic conditions within a road network. Addressing these short-comings of RFNs is an important direction for future work.

Second, data-abundance and data-sparsity co-occur in large vehicle trajectory sets: some road segments exhibit data abundance, whereas other road segments exhibit data sparsity. Within travel time and speed estimation—an important task for ITSs—this has given rise to two categories of approaches. Aggregation-based approaches assume data-abundance and can achieve high accuracy under such circumstances. Function-fitting approaches assume data-sparsity and can achieve good generalization performance for areas or times where there is little-to-no data available. This thesis proposes UniTE which aims to unify these approaches into a cohesive framework s.t. a function-fitting approach is applied in data-sparse conditions and an aggregation-based approach is applied in data-abundant conditions. UniTE achieves a high degree of data utilization since it used both at training time and at estimation time. An important direction for future work is the development of novel models that capitalizes on the synergy between function-fitting and aggregation within the UniTE framework.

Third, it is beneficial to process vehicle trajectory data that is in an easy-to-compress format to reduce storage and transmission costs. The thesis presents techniques for analysis driving behavior based on just the structure of map-matched vehicle trajectories represented as sequences of edges in a graph representation of a road network. This format is easier to compress since it contains no temporal information. The techniques allows mining of both driver intentions in the form of via-points and driving preferences in the form of a preference vector that quantifies the relative preference w.r.t. to different traversal costs, e.g., fuel consumption or travel time. Interestingly, an empirical study finds that temporal information stripped from the trajectories can be recovered to some extent. Exploring synergies between via-point identification and driver preference mining remain an important direction of future work.

References

[1] B. Yang, M. Kaul, and C. S. Jensen, “Using incomplete information for complete weight annotation of road networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1267–1279, 2013.

[2] J. Liu, G. P. Ong, and X. Chen, “Graphsage-based traffic speed forecasting for segment network with sparse data,” *IEEE Transactions on Intelligent Transportation Systems*, 2020, in online early access.

[3] L. Wei, Y. Wang, and P. Chen, “A particle filter-based approach for vehicle trajectory reconstruction using sparse probe data,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 5, pp. 2878–2890, 2021.

[4] H. Yuan, G. Li, Z. Bao, and L. Feng, “Effective travel time estimation: When historical trajectories over road networks matter,” in *Proc. of SIGMOD*, 2020, p. 2135–2149.

[5] X. Lin, Y. Wang, X. Xiao, Z. Li, and S. S. Bhowmick, “Path travel time estimation using attribute-related hybrid trajectories network,” in *Proc. of CIKM*, 2019, p. 1973–1982.

[6] B. Yang, J. Dai, C. Guo, C. S. Jensen, and J. Hu, “PACE: a PAth-CEntric paradigm for stochastic path finding,” *The VLDB Journal*, vol. 27, no. 2, pp. 153–178, 2018.

[7] H. Lu, D. Huang, Y. Song, D. Jiang, T. Zhou, and J. Qin, “ST-TrafficNet: A spatial-temporal deep learning network for traffic forecasting,” *MDPI: Electronics*, vol. 9, 2020, paper no. 1474.

[8] F. Barth, S. Funke, T. S. Jepsen, and C. Proissl, “Scalable unsupervised multi-criteria trajectory segmentation and driving preference mining,” in *Proc. of BIGSPATIAL*, 2020.

[9] T. S. Jepsen, C. S. Jensen, T. D. Nielsen, and K. Torp, “On Network Embedding for Machine Learning on Road Networks: A Case Study on the Danish Road Network,” in *Proc. of Big Data*, 2018, pp. 3422–3431.

[10] C. Chen, Y. Ding, X. Xie, S. Zhang, Z. Wang, and L. Feng, “TrajCompressor: An Online Map-matching-based Trajectory Compression Framework Leveraging Vehicle Heading Direction and Change,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 5, pp. 2012–2028, 2020.

[11] J. Muckell, P. W. Olsen, J.-H. Hwang, C. T. Lawson, and S. Ravi, “Compression of trajectory data: a comprehensive evaluation and new approach,” *GeoInformatica*, vol. 18, no. 3, pp. 435–460, 2014.

[12] T. Li, R. Huang, L. Chen, C. S. Jensen, and T. B. Pedersen, “Compression of uncertain trajectories in road networks,” *Proc. VLDB Endow.*, vol. 13, no. 7, p. 1050–1063, 2020.

[13] T. S. Jepsen, C. S. Jensen, and T. D. Nielsen, “Graph Convolutional Networks for Road Networks,” in *Proc. SIGSPATIAL'19*. ACM, 2019, p. 460–463.

[14] ——, “Relational Fusion Networks: Graph Convolutional Networks for Road Networks,” *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2020, in online early access.

[15] ——, “UniTE—The Best of Both Worlds: Unifying Function-Fitting and Aggregation-Based Approaches to Travel Time and Travel Speed Estimation,” 2021, submitted to *ACM TSAS*.

[16] H. Gao and H. Huang, “Deep attributed network embedding,” in *Proc. of IJCAI*, 2018.

[17] W. L. Hamilton, R. Ying, and J. Leskovec, “Inductive Representation Learning on Large Graphs,” in *Proc. of NIPS*, 2017, pp. 1024–1034.

[18] M. Buchin, A. Driemel, M. Van Kreveld, and V. Sacristán, “Segmenting trajectories: A framework and algorithms using spatiotemporal criteria,” *Journal of Spatial Information Science*, vol. 2011, no. 3, pp. 33–63, 2011.

Part II

Papers

Paper A

On Network Embedding for Machine Learning on Road Networks: A Case Study on the Danish Road Network

Tobias Skovgaard Jepsen
Aalborg University

Christian S. Jensen
Aalborg University

Thomas Dyhre Nielsen
Aalborg University

Kristian Torp
Aalborg University

This paper is published in the
Proceedings of the 2018 IEEE Internal Conference on Big Data,
pp. 3422–3431, 2018.

Received the Best Paper Award at the
3rd IEEE International Workshop on Big Spatial Data.

Abstract

Road networks are a type of spatial network, where edges may be associated with qualitative information such as road type and speed limit. Unfortunately, such information is often incomplete; for instance, OpenStreetMap only has speed limits for 13% of all Danish road segments. This is problematic for analysis tasks that rely on such information for machine learning. To enable machine learning in such circumstances, one may consider the application of network embedding methods to extract structural information from the network. However, these methods have so far mostly been used in the context of social networks, which differ significantly from road networks in terms of, e.g., node degree and level of homophily (which are key to the performance of many network embedding methods).

We analyze the use of network embedding methods, specifically node2vec, for learning road segment embeddings in road networks. Due to the often limited availability of information on other relevant road characteristics, the analysis focuses on leveraging the spatial network structure. Our results suggest that network embedding methods can indeed be used for deriving relevant network features (that may, e.g, be used for predicting speed limits), but that the qualities of the embeddings differ from embeddings for social networks.

Tobias Skovgaard Jepsen, Christian S. Jensen, Thomas Dyhre Nielsen, and Kristian Torp, 'On Network Embedding for Machine Learning on Road Networks: A Case Study on the Danish Road Network.' *In Proceedings of the 2018 IEEE International Conference on Big Data*, 2018, pp. 3422–343. DOI: 10.1109/BigData.2018.8622416.

The layout has been revised.

1 Introduction

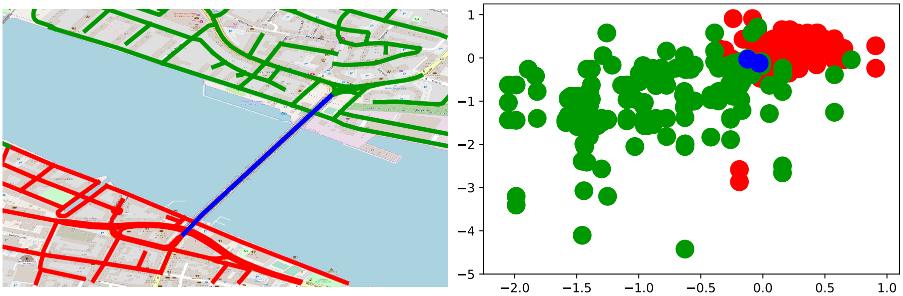
Road networks represent an important class of spatial networks and are an essential component of modern societal infrastructure. Road networks are associated with many important analysis tasks such as traffic flow and travel pattern analyses. In particular, many important road network tasks are supported by machine learning algorithms, including travel-time estimation [1, 2], traffic forecasting [3], and k nearest points-of-interest queries [4, 5], that require set of informative features to describe, e.g., the different road segments.

Solving road network analysis tasks is difficult since there is often little information available beyond the network structure itself. For instance, the Danish road network from OSM [6] contains only the network structure and up to two attributes characterizing each road segments: road category and speed limit. In addition, only 13% of the road segments have a speed limit label, even when augmented with data from Danish municipalities. This information sparsity makes it difficult to derive the features necessary for solving many road network analysis tasks. The road network structure is a potentially rich source of information, but it is not straight-forward to capture and utilize this often highly complex structure. For road network analyses, this typically involves explicit modeling of spatial correlations between adjacent road segments based on domain knowledge [1, 7, 8].

A road network is commonly modeled as a directed graph $G = (V, E)$, where each node $v \in V$ represents an intersection or the end of a road and each edge $(u, v) \in E$ represents a directed road segment that allows travel from u to v. Such graph representations makes *network embedding methods*—a class of feature learning methods for graphs—directly applicable for extracting structural information from road networks.

In network embedding, the goal is to learn a mapping (an *embedding*) that embeds nodes in networks into a d-dimensional vector space s.t. the node neighborhoods are preserved in the embedding space [9]. In other words, nodes are mapped to feature vectors that encode the structural information of the graph s.t. nearby nodes in the network are mapped to vectors that are near each other in the embedding space. For instance, Figure 1b shows that road segments north and south of the bridge in Figure 1a tend to cluster with other road segments from the same region. The road segments representing the bridge are somewhere in-between. Network embedding methods can extract the structural information in networks to supplement or replace attribute information if such information is low-quality, sparse, or unavailable.

The research in network embedding has thus far focused primarily on social, biological, and information networks [10–17]. Such networks differ significantly from road networks in terms of, e.g., structure, semantics, size,



(a)

(b)

Fig. 1: Illustration of (a) road segments in the Danish road network and (b) their feature vector representation generated by *DeepWalk* [10]. Colors indicate the region a road segment belongs to.

node degree, network diameter, and the amount of attribute information available. In addition, road networks may be disconnected due to inaccuracies in their spatial representation or the presence of islands, whereas, e.g., social networks are strongly connected. The effect of this disconnectedness on the embeddings is not obvious.

The differences between the types of networks studied in the network embedding literature, e.g., social networks, and road networks puts into question the suitability of using network embedding methods for road networks. We therefore formulate the following research question:

> *Are existing network embedding methods suitable for performing analysis tasks on road networks?*

To address this question, we conduct a case study: we evaluate an existing network embedding method empirically on road category classification and speed limit classification in the Danish road network. Only the road network structure is available beyond the road category and speed limit attributes which we wish to predict. *node2vec* [11] is therefore our network embedding method of choice since it relies solely on network structure while optimizing for the core property of neighborhood preservation in the embedding space; a property shared by most network embedding methods. Thus, node2vec is applicable when little or no attribute information is available in the network as is the case in our data set. We use the node2vec algorithm to learn embeddings of road segments in the Danish road network and subsequently feed these to a classifier to predict either road categories or speed limits.

Our key contribution is an empirical evaluation of network embedding methods for solving road network analysis tasks using our case study. In the network embedding literature, linear classifiers are commonly used to evaluate network embedding methods and can achieve high performance scores.

This suggests that network embedding methods tend to create embeddings that are linearly separable relative to the classification problem at hand, a highly-desirable property that makes it easier to apply machine learning algorithms. We see no reason to assume that these observations extend to the road segment classification tasks we consider. We therefore also investigate whether linear separability in the embedding space is present.

The success of most existing network embedding methods is due to the presence of strong homophily in many real networks [12], i.e., the tendency of connected nodes to be similar. Although node2vec relies primarily on homophily, it offers parameters that can emphasize structural equivalence in the embedding space to some extent. Structural equivalence differs substantially from homophily: two bridges may be considered structurally equivalent (or similar) despite not being connected and possibly far apart in the network. Based on the classification performance in our experiments, we interpret these parameters to gain insight into which type of similarity may be more appropriate for road network analysis tasks. We also evaluate the importance of other node2vec parameters.

Finally, the individual road categories and speed limits in our data set exhibits different degrees of homophily. We expect this to be problematic for subsequent classification on embeddings produced by a network embedding method. Consequently, we investigate the relationship between homophily and classification performance on individual classes.

Our evaluation shows that given the right choice of classifier and parameters, node2vec can achieve high macro F_1 scores of 0.57 for road category classification and 0.79 for speed limit classification; improvements by a factor of 8.3 and 11.5, respectively, over choosing the most frequent class in the training set. In addition, our experiments suggest that additional hyperparameter tuning for both node2vec and the classifier can result in even better classification performance. We also find that the class distribution in the embedding space reflects the class distribution in the road network, which, for the tasks we consider, results in a lack of linear separation in the embedding space. By interpreting the classification performance for different values of node2vec parameters, we find that structural equivalence may be a more important type of similarity for road networks than homophily. We also observed that a skew in the homophily of the classes results in a skew in classification performance on each class: highly homophilic classes achieve higher performances than less homophilic classes.

In summary, our contributions are as follows:

1. We empirically evaluate node2vec on two road network analysis tasks and demonstrate that it can achieve F_1 scores that are up to 11.5 times higher than choosing the most frequent class in the training set.

2. We show how the geometric distribution of classes in the network is

reflected in the embedding space, causing a loss of linear separability for the tasks in our case study.

3. We demonstrate the impact of different node2vec parameters on classification performance, and show that emphasizing structural equivalence in the embedding space results in higher classification performance.

4. We show that the reliance on neighborhood preservation results in a skew in classification performance towards favoring strongly homophilic classes.

The rest of the paper is structured as follows. In Section 2, we discuss related methods that embed road network intersections and road segments and have been developed in parallel with network embedding methods. In Section 3 we give the necessary background information on network embedding methods. In Section 4 we evaluate the node2vec embedding. Finally, we represent our conclusions and discuss how we expect our findings to generalize to other network embedding methods and road network analysis tasks in Section 5.

2 Related Work

Embedding of road network intersections and segments is largely unexplored. To the best of our knowledge, no previous work exists that investigates the application, adaptation, or extension of (general) network embedding methods to embed road network intersections and segments. However, a few road network embedding methods—designed specifically for embedding road network intersections and segments—have been developed in parallel with (general) network embedding methods. We proceed to review these methods.

Shahabi, et al. [4] use embeddings of intersections to improve the accuracy of k-nearest neighbor queries that find the k nearest points of interest for moving vehicles. Specifically, they embed intersections s.t. the shortest-path distance between two intersections is better approximated by the L_{inf} distance between the embedded intersections than by the Euclidean distance between their geographical points. Liu et al. [5] further extend this approach to support private k-nearest neighbor queries, where the exact location of a vehicle is hidden. As in our setting, only the road network structure is available, but we investigate whether network embedding methods may be applicable for a wider variety of road network tasks whereas [5] aim to produce vectors that preserve a distance relationship between intersections for a specific task.

Road2Vec [3] learns embeddings of road segments that capture traffic interactions. A traffic interaction between two road segments happens when the road segments co-occur within some distance of each other in a vehicle GPS trajectory. Using such co-occurrences, road2Vec assigns similar vector representations in the embedding space to road segments that interact frequently. Road2Vec requires a set of vehicle trajectories that covers all road segments in the road network to be able to learn meaningful similarities between road segments. This cannot be expected in general [7] and in our setting GPS trajectories are not available.

Another road segment embedding method, by Fruensgaard and Jepsen [2], relies on attribute information from nodes and edges in random walks to generate an embedding. Rather than capturing traffic interactions, this method captures a notion of structural equivalence between road segments based on the attributes of their surrounding road segments. They demonstrate that their method achieves superior performance on trip travel time prediction tasks compared to approaches based on traditional feature engineering with similar development effort [2]. This method and Road2Vec are quite similar to the network embedding method, node2vec, that we use in our case study. Like node2vec, these methods rely on samples of neighborhoods from the road network, either in the form of GPS trajectories or walks, and use techniques from natural language processing to produce the embeddings. However, contrary to our setting, this method assumes the presence of rich attribute information.

3 Network Embedding

We proceed to give the relevant background in network embedding. We first give a general introduction to network embedding, and describe node2vec in detail. We then briefly review other selected methods.

3.1 A General Introduction

Let $G = (V, E)$ be the graph representation of a network. A network embedding is a function $\phi \colon V \cup E \to \mathbb{R}^d$ that maps network elements to d-dimensional vectors. Network embedding methods often aim to learn a node embedding, but an edge may be embedded by aggregating the embeddings of the edge's incident nodes [11]. For instance, an embedding of an edge (v_1, v_2) can be found by concatenating the embeddings of its source and target nodes: $\phi(v_1, v_2) = [\phi(v_1)\phi(v_2)]$.

Deriving edge embeddings from node embeddings is useful for link prediction in social networks, where the task is to predict the existence of a missing edge [11]. If all edges are known, an edge embedding can instead be produced by learning a node embedding for the dual graph representation of the network. In both cases, network embedding methods can be used to embed road segments.

Network embedding methods aim to map similar network elements to similar vector representations in the embedding space by optimizing an objective function that specifies the notion of similarity. In particular, network embedding methods find the node embedding ϕ that maximizes the following objective [10–13]:

$$\sum_{v \in V} \log \Big(\Pr(N(v) \mid \phi(v)) \Big), \tag{A.1}$$

where $N(v)$ is the neighborhood of node v and $\Pr(N(v) \mid \phi(v))$ is the probability of observing the neighborhood $N(v)$ of v given its feature representation $\phi(v)$. The probability $\Pr(N(v) \mid \phi(v))$ is computed using the softmax function [11] or some approximation thereof. Other similar objectives have also been considered [14–16, 18].

The notion of neighborhood is not restricted to immediate neighbors, and many methods sample neighborhoods using random walks [10–12, 18]. In such cases, the notion of node neighborhood is defined in terms of a set of walks across the node s.t. given a walk $(v_1, \ldots, v_n)$, the neighborhood of a node v_i is the c preceding and succeeding nodes in the walk $\{v_{i-c}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+c}\}$, where c is the context size. Neighborhood sampling using random walks can scale to very high-degree networks; examples include social networks where nodes have thousands or even millions of neighbors[1].

Maximizing Equation A.1 results in an embedding that is optimized to preserve node neighborhoods (according to $N(v)$) in the embedding space [11, 12]. Intuitively, Equation A.1 suggests that the similarity between two nodes u and v in the embedding space is proportional to the size the overlap of their neighborhoods. For such an embedding to be useful, the underlying network must exhibit *homophily*: the tendency of network elements to be connected to similar network elements [12]. From the perspective of a supervised learning task, homophily means that neighboring nodes are likely to have the same label. An embedding that preserves node neighborhoods places neighboring nodes close in the embedding space and therefore produces useful feature vectors for subsequent machine learning under the homophily assumption.

We anticipate that producing a good embedding for road networks while relying on neighborhood preservation is difficult for three reasons. First, a road network is a spatial construct and inaccuracies in the spatial representation of a road network may result in false or missing edges in the graph representation. Next, countries such as Denmark have islands that may result in disconnected subgraphs. The effect of such subgraphs on the training of the

[1] The YouTube account on Twitter has 70.6 million followers at the time of writing.

network embedding is not immediately obvious, but it may add noise to the training procedure, resulting in a reduced degree of neighborhood preservation. Finally, island nodes and main land nodes are not similar according to Equation A.1 since they are neither neighbors nor share neighbors.

3.2 node2vec

We use the embedding method node2vec in our case study. We first discuss node2vec's training objective and then proceed to look at the flexible sampling strategy employed by node2vec.

First, node2vec optimizes Equation A.1 directly and uses random walks to represent node neighborhoods. For each node v in a graph, node2vec samples r walks of maximum length l starting from v s.t. the next node visited in the walk is chosen at random among the neighbors of the last node in the walk. Therefore, the neighborhood of a node v is distributed over the walks. To illustrate this, we rewrite Equation A.1 to reflect this distributed neighborhood in Equation A.2. Note that Equation A.2 assumes that the neighborhoods of a node v w.r.t. each walk are conditionally independent given $\phi(v_i)$.

$$\sum_{W \in \mathcal{W}} \sum_{v_i \in W} \log \Big(\Pr \left(N^W(v_i) \mid \phi(v_i) \right) \Big) \tag{A.2}$$

Here, $\mathcal{W}$ is a set of $r \cdot |V|$ walks $W = \{v_1, \ldots, v_k\}$ s.t. $k \leq l$ and $N^W(v_i) = \{v_{i-c}, \ldots, v_{i-1}, v_{i+1}, \ldots, v_{i+c}\}$ is the neighbors of node v_i with respect to walk W. The context size c adjusts the number of preceding and succeeding nodes in the walk to consider the neighbors of v_i. In general, c should be selected s.t. $2c << l$ to avoid frequently padding $N^W(v_i)$ with "null" nodes.

Next, node2vec samples w walks with a maximum length of l using a second-order biased random walk, where w and l are hyperparameters of node2vec. Given that the random walk has just traversed an edge (v_{i-1}, v_i) in an unweighted graph and now resides at node v_i, the probability of visiting a node v_{i+1} is [11]:

$$\Pr(v_{i+1} \mid v_i, v_{i-1}) = \frac{1}{Z} \cdot \begin{cases} \frac{1}{p} & \text{if } d(v_{i-1}, v_{i+1}) = 0 \\ 1 & \text{if } d(v_{i-1}, v_{i+1}) = 1 \\ \frac{1}{q} & \text{if } d(v_{i-1}, v_{i+1}) = 2 \\ 0 & \text{if } (v_i, v_{i+1}) \notin E \end{cases}, \tag{A.3}$$

Here, Z is a normalization constant and $d(u, v)$ returns the distance between two nodes u and v. p and q are hyperparameters that change the behavior of the walk to behave as a Breadth-First Search (BFS), Depth-First Search (DFS), or something in-between. A BFS emphasizes homophily and a DFS emphasizes structural equivalence as the type of similarity to capture in the embeddings [11].

The *return parameter* p adjusts the probability of revisiting the previous node v_{i-1} in the walk and restricts the number of different nodes visited in the search. Low values of q is equivalent to restricting the search depth in a DFS and equivalent to restricting the number of neighbors to explore in a BFS. The *in-out parameter* q adjusts the probability of visiting different neighbor of v_{i-1}. In effect, q adjust the behavior of the walk to become more BFS-like at high values and more DFS-like at low values.

Although node2vec can capture structural equivalence to some extent, even using an actual DFS to sample neighborhoods results in neighborhoods that can include nodes up to a maximum distance of c. Thus, the structural equivalence emphasized by the walks is local to the area of the network from which it is sampled. This makes the walks unable to capture structural equivalences between, e.g., bridges that are far apart in the network, and thus node2vec still primarily relies on homophily in networks with large diameters such as country-sized road networks.

The embedding function ϕ is generated using a single-layer neural network. A node v_i is embedded in the d hidden units of the (hidden) embedding layer. The resulting d-dimensional embedding is then used to predict v_i's neighbors (w.r.t. a walk) at the output layer by using the softmax function (or some approximation thereof) to compute the probability $\Pr(N^W(v_i) \mid \phi(v_i))$ in Equation A.2.

3.3 Other Approaches

The discussion of network embedding methods has thus far been relatively focused. For completeness, we briefly review other approaches to network embedding; however, we note that all of the following methods to some extent preserves node neighborhoods as in Equation A.1. Fundamentally, network embedding methods differ primarily in either the choice of neighborhood function or in how (and if) they incorporate node or edge attributes in the embedding.

Alternative Neighborhood Function

So far, a node neighborhood has been expressed as all nodes within c hops of a node or some subset thereof. As discussed previously, this leads to a neighborhood preserving embedding. The *struc2vec* embedding method [12] changes the neighborhood sampling procedure by sampling walks from a multi-layered graph. A node v in a layer k, is connected by an edge to all nodes at exactly distance k; hence each layer represents the neighborhood of v at different "zoom" levels. The walk can choose to stay at the current layer or to proceed to a higher layer depending on a heuristic designed to preserve structural similarity. This enables struc2vec to effectively skip nodes in the walk. The worst-case time and space complexities of struc2vec are $\mathcal{O}(|V|^3)$ and $\mathcal{O}(|V|^2)$, respectively. These complexities can be improved significantly through various optimization methods, but continues to be super-linear [12], which hinders its applicability to large road networks.

Incorporation of Attributes

Network embedding methods that incorporate attributes use them as input to the embedding method [13, 16] and possibly include a sub-objective in the objective function that encourages nodes or edges with similar attributes to obtain similar vector representations in the embedding space [14, 15, 18]. This allows such approaches to compute a notion of similarity even between disconnected or distant nodes. However, these methods still aim to preserve node neighborhoods in the embedding space to some extent.

4 Experimental Study

To investigate the suitability of network embedding methods for road network analysis, we evaluate the node2vec method on two road segment classification tasks and report the results.

The existing embedding literature commonly uses linear classifiers to emphasize the quality of the embedding over the complexity of the classifier, and typically such classifiers achieve high classification performance. This suggest that network embedding methods tend to make the classification problems in the literature linearly separable in the embedding space. Given that such methods are typically applied to, e.g., social networks, that exhibit characteristics different from road networks, it is not clear whether linear separability in the embedding space extends to road networks. We therefore investigate whether this property is present for the classification tasks that we consider.

Network analysis tasks typically depend on homophily or structural equivalence as the notion of similarity [11]. To investigate which type of similarity is appropriate for road network analysis tasks, we exploit the interpretability of the node2vec random walk parameters p and q, as discussed in Section 3.2, by investigating the impact of these parameters on classification performance. We also investigate the impact of the node2vec architecture parameters, the dimensionality d and context size c, to gain insight into how changes in the node2vec architecture influences classification performance.

The success of most existing network embedding methods is due to the presence of strong homophily in many real networks [12]. In our data set, the individual classes exhibit different degrees of homophily, which suggest that this is problematic for network embeddings methods. We therefore investigate the relationship between homophily and classification performance on individual classes.

4.1 Data Set

We have extracted the spatial representation of the Danish road network from OSM [6]. We represent the road network as a directed multigraph, where each node represents an intersection or the end of a road and each edge (u, v) represents a road segment that enables travel from node u to node v. This yields a graph consisting of $583,816$ nodes and $1,291,168$ edges.

The data from OSM contains two road segment attributes, road category and speed limit. We further augment the data set with additional speed limit information from the Danish municipalities of Aalborg and Copenhagen. This results in $1,291,168$ road segments (i.e., all edges) labeled with one of 9 road categories and $163,043$ road segments (~13% of all edges) labeled with one of 10 speed limits. We also note that the speed limits are not distributed evenly geographically: speed limits in major cities are over-represented in the data.

As discussed in Section 3, network embedding methods are suited for networks that are homophilic w.r.t. the attributes of interest. We therefore measure the homophily for both the road category and speed limit attributes.

We measure the homophily of an attribute value a in a directed network $G = (V, E)$ as *the empirical probability that an edge $e_1 = (u, v)$ is adjacent to an edge $e_2 = (v, w)$ that has attribute value $A(e_2) = a$ given that e_1 has attribute value $A(e_1) = a$*, i.e.,

$$\begin{aligned} H_G^c &= \Pr(A(v,w) = a \mid A(u,v) = a) \\ &= \sum_{(u,v)\in E} \sum_{(v,w)\in E} \frac{\mathbb{1}[A(u,v) = A(v,w)]}{Z}, \end{aligned}$$

where $A(v_1, v_2)$ is the attribute value of an edge (v_1, v_2) and Z is a normalization constant.

We compute the homophily of network G with regards to an attribute $A = \{a_1, \ldots, a_m\}$ as follows.

$$H_G^A = \sum_{a\in A} \frac{H_G^a}{|A|}$$

We summarize the data set statistics in Table 1. As can be seen, there is a homophily of $H_G^L = 72.3\%$ for road categories and $H_G^L = 75.8\%$ for speed limits. A notable outlier is the road category "Motorway Approach/Exit" which has a homophily of 36.8%.

Table 1: Data Set Statistics for Road Categories and Speed Limits

	Class	*Homophily*	*Frequency*
Road Categories			
	Residential	90.4%	**570,820 (44.2%)**
	Service	72.7%	278,985 (21.6%)
	Unclassified	78.0%	257,726 (20.0%)
	Tertiary	70.2%	103.830 (8.04%)
	Secondary	70.6%	52,021 (4.03%)
	Primary	71.7%	22,255 (1.72%)
	Motorway	78.2%	2,236 (0.173%)
	Motorway Approach/Exit	**36.8%**	1,749 (0.135%)
	Trunk	81.7%	1,546 (0.120%)
Mean/Total		72.3%	1,291,168 (100%)
Speed Limits			
	50	82.2%	**85,377 (52.4%)**
	80	73.1%	37,750 (23.2%)
	40	79.7%	11,830 (7.26%)
	60	64.9%	10,112 (6.20%)
	30	78.4%	9,093 (5.58%)
	70	63.2%	4,481 (2.75%)
	20	80.7%	1,383 (0.848%)
	110	70.5%	1,103 (0.677%)
	90	72.9%	1,087 (0.664%)
	130	71.5%	827 (0.507%)
Mean/Total		75.8%	163,043 (100%)

4.2 Experiment Design

We use the node2vec [11] embedding to classify road categories and speed limits as follows.

1. We first sample $r = 10$ walks, starting from each intersection in the road network, with a maximum length of $l = 80$ using the biased random walk in Equation A.3 parameterized by the return parameter p and the in-out parameter q.
2. We then learn the embedding using these walks as input.
3. Finally, we train a classifier for each of the road segments classification tasks using these embeddings.

To investigate the linear separability in the embedding space, we consider both a linear and a non-linear classifier:

- a one-vs-rest logistic regression model as in the original node2vec paper [11], and
- a random forest classifier [19], a powerful ensemble model, with 10 decision trees.

After learning the embedding, we generate a feature vector for each road segment by concatenating the embeddings of its source and target nodes as discussed in Section 3. For both road category and speed limit classification, we randomly choose 50% of the labels without replacement for training and use the remaining 50% for testing. To deal with the large class imbalance in our data set (see Table 1), we randomly over-sample all classes with replacement s.t. the frequency of all classes in the training set match the frequency of the majority class. We then train road category and speed limit classifiers using these feature vectors to represent each labeled road segment in the over-sampled training set. Finally, we evaluate the classifier on the two road segment classification tasks using the macro F_1 score, which is commonly used in the network embedding literature [10, 11, 13, 18]. The macro F_1 scores punishes poor performance equally across all classes, rather than rewarding good performance on the very frequent classes in our imbalanced data set.

To find the best node2vec parameter configuration for each of the classifiers, we learn node2vec embeddings with different configurations of return parameter p, the in-out parameter q, the context size c, and the dimensionality d. We explore configurations based on the following possible parameter values: $p \in \{0.25, 0.5, 1, 2, 4\}$, $q \in \{0.25, 0.5, 1, 2, 4\}$, $c \in \{1, 5, 10, 15, 20, 25, 30\}$, and $d \in \{64, 128, 256\}$. We use the baseline values of return parameter $p = 1$, in-out parameter $q = 1$, and context size $c = 10$. We then explore different values for p, q, and c at different values of d while keeping the other two parameters at their baseline values. Although use of these parameter configurations does not result in an exhaustive search of the parameter space, they do allow us to evaluate the impact of each parameter on classification performance. Finally, the c and d parameters do not influence the walk sampling procedure, and we therefore reuse walks for parameter configurations that have the same p and q values.

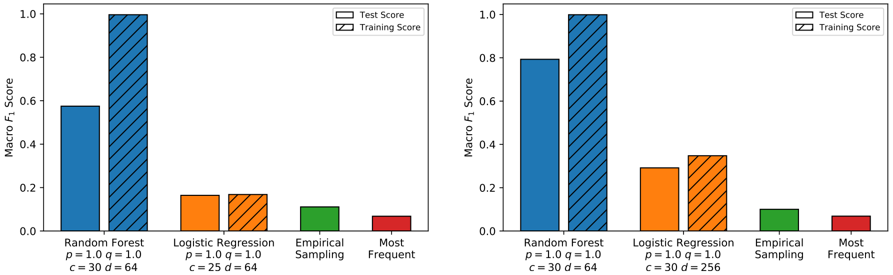
To establish baseline performances for evaluation, we use two simple classifiers that classify road segments using only the statistics of the training set:

- *Most Frequent*: Always predicts the most frequent class in the training set.

- *Empirical Sampling*: Draws a class at random from the empirical distribution of classes in the training set.

4.3 Road Segment Classification

We show the performance of the best performing node2vec parameter configuration for road category and speed limit classification using logistic regression and random forests (with baselines for comparison) in Figure 2. As can be seen, the choice of classifier has a large impact on classification performance. The macro F_1 score using a random forest is 0.57 for road category classification and 0.79 speed limit classification. This is roughly three times



(a) Road Category Classification.

(b) Speed Limit Classification.

Fig. 2: Macro F_1 scores on the training and test sets for the best performing embedding for each classifier with baselines for comparison on (a) road category classification and (b) speed limit classification.

higher than the score achieved using logistic regression, and 8.3 and 11.5 times higher than the two baselines, respectively. We expect that these results can be improved by choosing appropriate p and q parameter values, as we shall discuss in Section 4.5. In addition, the random forest achieves a near-perfect score on the training set for both tasks, which suggests that it is over-fitting and could conceivably achieve even higher performance by tuning its hyperparameters. On the other hand, the logistic regression model achieves, roughly equal performance on the training and test sets for both classification tasks, and is only marginally better than the Empirical Sampling baseline on road category classification, as shown in Figure 2a.

4.4 Linear Separability

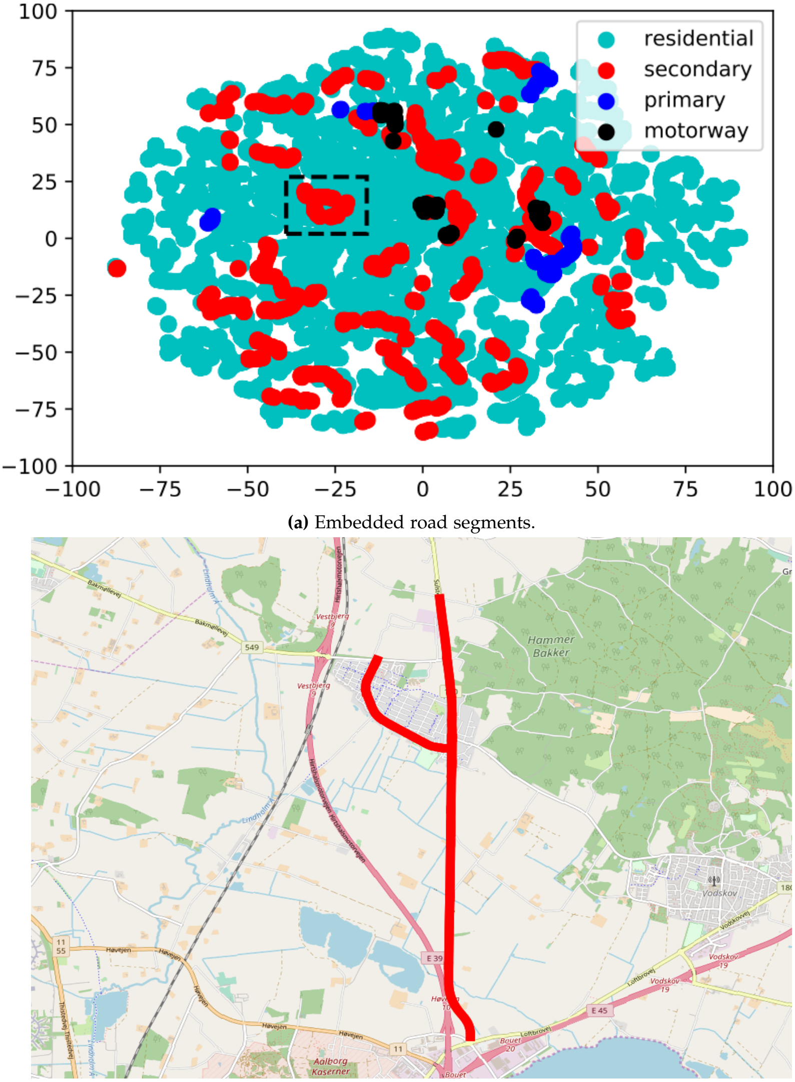
It is not surprising that the random forest model outperforms the logistic regression model given that the random forest uses advanced ensemble learning techniques, such as boosting. However, the low score on both training and test sets shown in Figure 2 suggests that the logistic regression model is both unable to fit the training data and generalize to the test data. This suggests that the logistic regression model is unable to find good linear decision boundaries and thus that the classes are not linearly separable in the embedding space.

We investigate the linear separability in the embedding space in detail by embedding the road network of Aalborg Municipality, Denmark, using node2vec with baseline parameters and visualizing the position of the road segments in the embedding space by reducing the dimensionality of their feature vectors. We use Barnes-Hut t-SNE [20] to project the feature vectors into two dimensions and plot them in Figure 3. Each point in the plot represents a directed road segment, and is colored according to its road category. For illustrative purposes, we cover only four of the road categories in the plot.

As can be seen in Figure 3a, road categories are not well-separated in the two-dimensional t-SNE projection. The residential road segments are scattered almost uniformly over the embedding space, and the remaining categories are scattered in several smaller clusters. Each such cluster corresponds to a homophilic neighborhood (or area) in the road network. For instance, the cluster of secondary road segments highlighted in Figure 3a corresponds to the 57 road segments highlighted in red in Figure 3b. These scattered clusters suggest that it is difficult to find good linear decision boundaries for the logistic regression model, which is further supported by its poor performance on both the training and test sets as shown in Figure 2.

The lack of separation in the embedding space reflects the lack of separation in the road networks due to the neighborhood preserving properties of the embedding method: for example, several clusters of secondary segments can be found in different areas of the road network. Although we have focused the discussion on road categories, speed limits show a similar data distribution.

Given that logistic regression is unable to find linear decision boundaries in the embedding space that are competitive with the random forest classifier, we focus on the random forest classifiers for our experimental results in the remainder of the paper.



(a) Embedded road segments.

(b) Spatial representations of the framed cluster in Figure 3a.

Fig. 3: Two-dimensional t-SNE projections of the road segment embeddings in Aalborg Municipality.

4.5 Homophily or Structural Equivalence

Next, we examine how the return parameter p and the in-out parameter q influences classification performance.

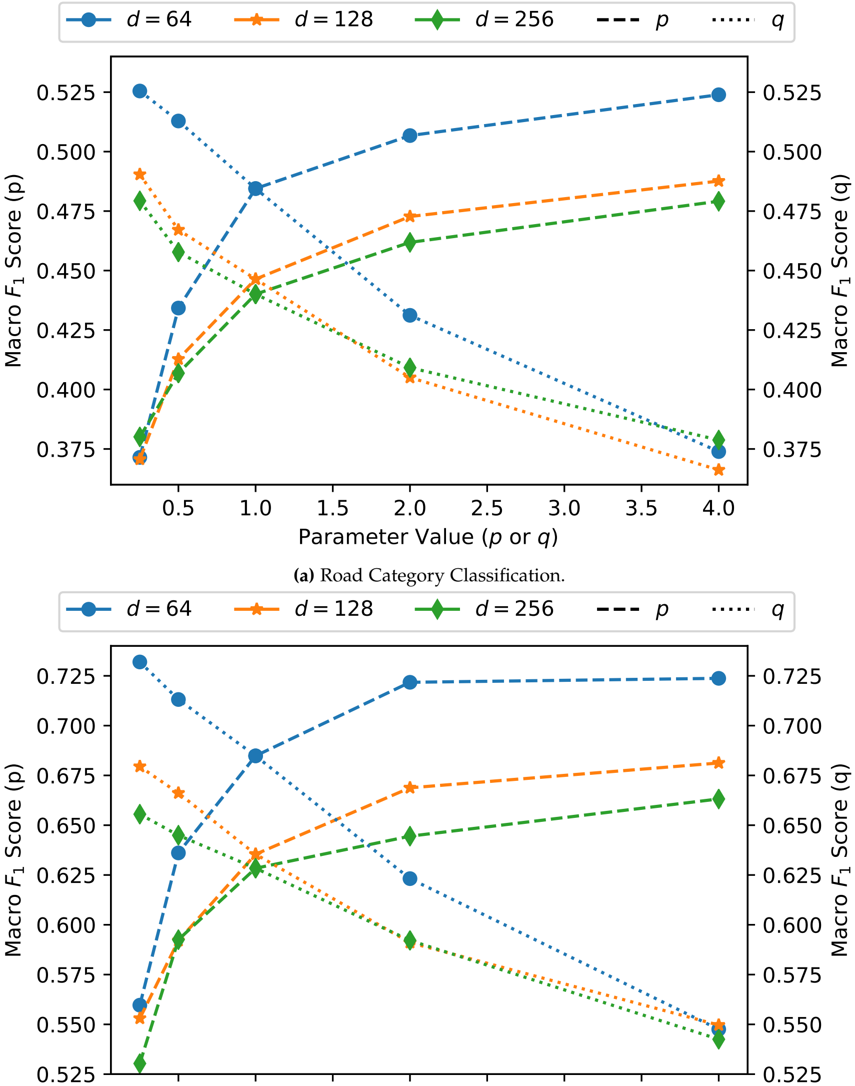
Recall from Section 3.2 that low values of p increases the probability of revisiting the previous node in the walk and that the parameter q makes the walk behave like a DFS for low values of q and like a BFS for high values of q. Intuitively, p controls the breadth or depth of the search in terms of how many different nodes are visited, while q controls to which extent the random walk behaves more like a BFS or a DFS. In addition, recall that a BFS-like walk emphasizes homophily in the embedding space and a DFS-like walk emphasizes structural equivalence (but restricted by the neighborhood defined by the walk). Thus, this structural equivalence is local, in the sense that structural equivalence between, e.g., bridges in different parts of the country cannot be captured in networks with large network diameter. However, node2vec may still provide insight into the appropriate type of similarity for the road segment classification tasks.

We plot the classification performance for the random forest classifier for different values of p and q on the two road segment classification tasks in Figure 4. As shown in the figure, classification performance is highest at high values of p and low values of q, regardless of the dimensionality. The high performance at high values of p suggests that exploring more nodes in the walk sampling procedure is beneficial. The high performance at low values of q suggests that a DFS-like neighborhood exploration is superior to a BFS-like neighborhood exploration. We do not reach a saturation point for the q values which suggest that the more DFS-like the walk, the better. Thus, our results suggest that structural equivalence should be emphasized over homophily in the embedding space.

We make the additional observation that p and q adversely affect each other in Equation A.3: a high p value reduces the probability of visiting a node at distance two from the previously visited node and a high q value reduces the probability of returning to a previously visited node. We therefore propose that the ratio between these parameters is more important than their absolute values.

For the parameter configurations we explore in our experiments, a ratio of, e.g., $\frac{p}{q} = 4$ can occur for values $p = 1$ and $q = 0.25$ or values $p = 4$ and $q = 1$. We therefore plot the data points for both the case where $p > q$ and the case where $q > p$ by taking the mean of their associated macro F_1 scores; see Figure 5. We also investigated whether $p > q$ should be preferred over $q > p$ for each ratio, but we found no trend for the values that we examined to suggest that one combination of p and q values should be preferred over the other given that they have the same $\frac{p}{q}$ ratio.

As shown by the figure, a larger $\frac{p}{q}$ ratio results in a higher classifica-



(a) Road Category Classification.

(b) Speed Limit Classification.

Fig. 4: The effects of p (dashed) and q (dotted) on (a) road category classification and (b) speed limit classification using a random forest classifier.

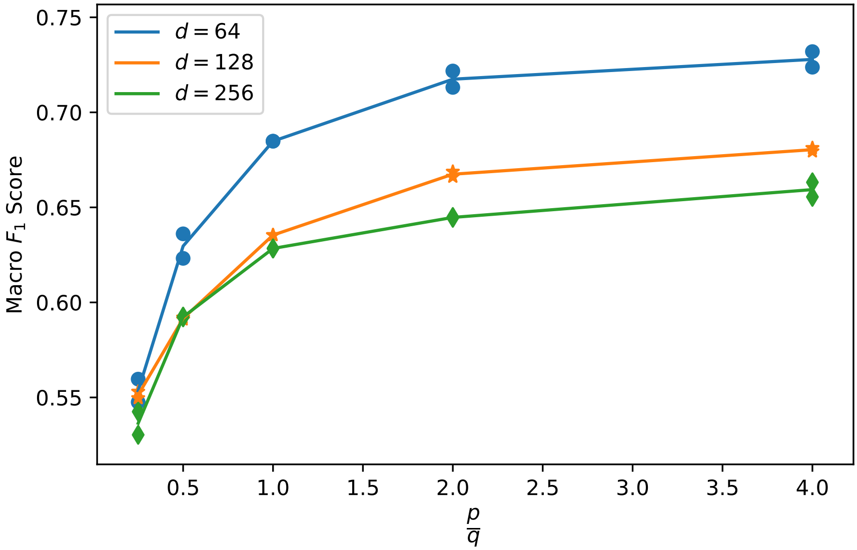


Fig. 5: The relationship between $\frac{p}{q}$ and classification performance on the speed limit classification task using a random forest classifier.

tion performance on the speed limit classification task. We observe the same pattern on road category classification. This suggests that emphasizing structural equivalence in the embedding space is more important than emphasizing homophily for the tasks we consider which is consistent with our previous observations on the values of p and q.

4.6 Architectural Parameters

We also investigate the impact of context size c and dimensionality d on classification performance. These parameters adjust the amount of neighborhood information available and the number of weights available for node2vec.

The context size c and the return parameter p serve somewhat similar roles since they both control the number of different nodes included in the neighborhood: large values of c include more nodes from the walks in the node neighborhood whereas small values of p makes it less likely that we revisit nodes and thus increases the number of different nodes that occur in the neighborhood of a node. We therefore expect that larger context sizes results in higher performance.

We plot the performance of the random forest classifier on speed limit classification at different context sizes in Figure 6. As the figure shows, the performance is initially low with a context size of $c = 1$, but quickly increases as c increases until performance starts to flatten at $c = 15$. We observe the

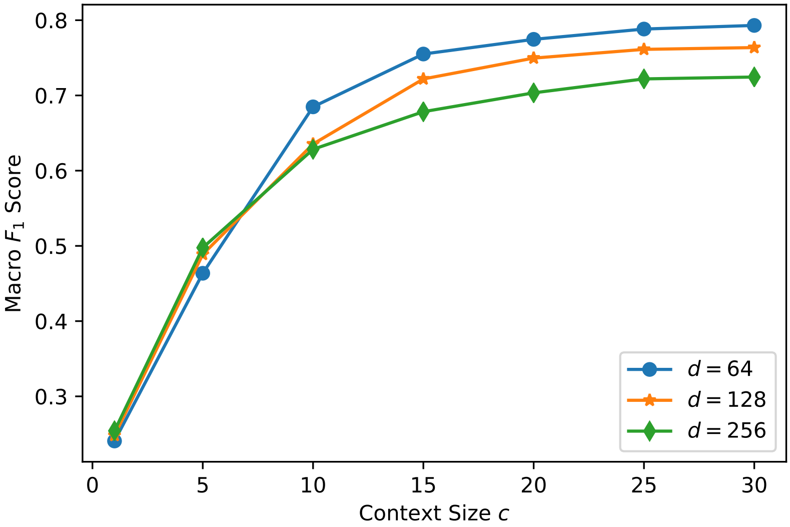


Fig. 6: Classification performance at different context sizes on the speed limit classification task using a random forest classifier.

same trend on road category classification.

Although our results indicate that larger context sizes yield superior or equivalent performance, we expect that very large context sizes can introduce noise. This could happen if the context size is sufficiently large that otherwise different intersections (and road segments) in different parts of the network have large overlaps between their neighborhoods. From the perspective of the optimization problem in Equation A.1, this renders them nearly indistinguishable.

Dimensionality does not influence our conclusions regarding the impact of the other node2vec parameters on classification performance. For the random forest classifier, the lowest dimensionality $d = 64$ performs up to 20% better than $d = 128$ and $d = 256$ depending on the choice of p and q, as shown in Figure 4. $d = 64$ is also the highest performing dimensionality of d at different context sizes, except at a context size of $c = 5$, as shown in Figure 6. We suspect this is because the random forest is more prone to overfit the training set for larger values of d.

4.7 Homophily and Classification Performance

As shown in Table 1, there is a skew in the class homophilies for road categories. In particular, the category ”Motorway Approach/Exit” exhibits substantially lower homophily than the other road categories. We observe that speed limits exhibit both higher homophily on average and higher classifica-

Table 2: Homophily and F_1 scores for each road category using the best random forest classifier.

Class	*Homophily*	F_1 *Score*
Residential	**90.4%**	**0.83**
Trunk	81.7%	0.67
Motorway	78.2%	0.62
Unclassified	78.0%	0.62
Service	**72.7%**	**0.56**
Primary	**71.7%**	**0.57**
Secondary	70.6%	0.54
Tertiary	70.2%	0.52
Motorway Approach/Exit	**36.8%**	**0.25**

tion performance in our experiments. As discussed in Section 3, the network embeddings assume homophily in the network and this assumption is the primary driver for their success. We therefore expect that the random forest classifier achieves higher performance on road categories such as ″Residential″ that exhibits strong homophily than road categories such as ″Motorway Approach/Exit″ that exhibits weaker homophily.

As shown in Table 2, there is a near-perfect correspondence between the road categories as ordered by their homophily and macro F_1 scores. ″Residential″ and ″Motorway Approach/Exit″ has the highest and lowest F_1 scores of 0.83 and 0.25, respectively. The only discrepancy is between the ″Service″ and ″Primary″ categories, where ″Primary″ has slightly higher F_1 score (0.57) than ″Service″ (0.56), despite having a slightly lower homophily. This discrepancy may be due to randomness in the training process or disparity between the notion of neighborhoods employed in the homophily measure: we measure homophily based only on the immediate neighbors of a road segment, but the embedding is trained using random walks to represent the node neighborhoods that do not correspond to immediate neighbors. We observe the same pattern for speed limits.

In conclusion, the results in Table 2 suggests that the classification performance depends strongly on the homophily in the network and that classifier performance is skewed in favor of classes that exhibit strong homophily.

5 Discussion, Conclusion, and Future Work

We have investigated the suitability of network embedding methods for machine learning on information-sparse road networks by evaluating an existing network embedding method, node2vec, for road category and speed limit classification in the Danish road network.

We have shown that it is possible to achieve macro F_1 scores of 0.57 and 0.79 on road category classification and speed limit classification, respectively. Depending on the task, these scores are between 8.3 and 11.5 times higher than guessing the most frequent class in the training set. Furthermore, our results suggests that this performance can be increased further by appropriate parameter tuning of both node2vec and the used classifier. This suggests that network embedding methods may be useful at extracting structural information from not only social networks, but also road networks.

Some degree of linear separability is implied by the prolific use of linear classifiers in the network embedding literature. We therefore investigated linear separability in the embedding space for our tasks. For both classification tasks, we found that we were unable to fit the training set or generalize to the test using a linear classifier. By visualizing the embedding space, we found that the members of each class are distributed across several scattered clusters in the embedding space, which reflects the geometric distribution of classes in the network, and suggests that it is difficult to find good linear decision boundaries in the embedding space.

We explored the classification performance for different node2vec parameter configurations. We found that the impact of the return parameter p and in-out parameter q on classification performance suggests that structural equivalence, as opposed to homophily, is the more appropriate type of similarity. We also evaluated the other parameters, and found that the dimensionality of the embeddings becomes increasingly more important at high values of p and low values of q, with a preference for low dimensionality for the non-linear classifier and a preference for high dimensionality for the linear classifier. In addition, it is preferable to include more nodes in the node2vec neighborhood function by using large context sizes, although we expect that large context sizes introduce noise in the embeddings.

Finally, we investigated the relationship between the class homophily and classification performance. We found that classification performance is better on classes with high homophily than classes with low homophily, and which we propose is a result of node2vec's neighborhood preservation in the embedding space and the corresponding homophily assumption.

We expect that our findings generalize to other network embedding methods. Specifically, node2vec is a neighborhood preserving network embedding method, and, as demonstrated in our experiments, it tends to achieve higher performance on tasks and classes with higher homophily. As discussed in Section 3, network embedding methods preserve node neighborhoods in the embedding space as either the main objective or as a sub-objective in combination with attribute information. We therefore expect that we can achieve similar or better results using different network embedding methods.

The tasks that we have examined exhibit strong homophily. In addition, the classes in our data set are scattered in clusters in different areas of the road network. We therefore expect that our findings generalize to tasks that are similar w.r.t. homophily and data distribution in the network. For instance, driving speeds correlate between adjacent segments [1], and driving speeds in, e.g., two distant cities are often similar.

We propose that further attention should given to embedding methods for road networks with sparse information. To the best of our knowledge, no network embedding method exists that can leverage the spatial information of road networks. In our data set, road segments may be as short as a few meters for road segments in roundabouts and as a long as several kilometers for motorways. This means that, e.g., the context size has different meaning for different intersections. Future work in network embedding should therefore explore the utilization of the spatial information.

We also found that it is hard to predict the road category of motorway approaches and exits due to their low homophily. This particular road category has low homophily because road segments of this category connect different categories of road segments, e.g., motorways to highways, or vice versa. Thus, motorway approaches and exits play a particular role in the road network, and structural equivalence as a notion of similarity may be much better suited for this road category. On the other hand, residential road segments are strongly homophilic and appear much better suited for homophily as a notion of similarity. This suggests that there is a need for road network embedding methods that can capture both structural equivalence and homophily, and balance these notions of similarity for each class.

Acknowledgments

This research was supported in part by the DiCyPS project and by grants from the Obel Family Foundation and the Villum Foundation. We thank the OpenStreetMap (OSM) contributors, without whom this work would not have been possible. Map data copyrighted OpenStreetMap contributors and available from `https://www.openstreetmap.org`.

References

[1] J. Zheng and L. M. Ni, "Time-dependent trajectory regression on road networks via multi-task learning," in *Proc. of AAAI*, 2013, pp. 1048–1055.

[2] M. Fruensgaard and T. S. Jepsen, "Improving cost estimation models with estimation updates and road2vec: a feature learning framework for road networks," Master's thesis, Aalborg University, 2017.

[3] K. Liu, S. Gao, P. Qiu, X. Liu, B. Yan, and F. Lu, "Road2Vec: Measuring Traffic Interactions in Urban Road System from Massive Travel Routes," *IJGI*, vol. 6, no. 11, 2017, article No. 321.

[4] C. Shahabi, M. R. Kolahdouzan, and M. Sharifzadeh, "A road network embedding technique for k-nearest neighbor search in moving object databases," *GeoInformatica*, vol. 7, no. 3, pp. 255–273, 2003.

[5] F. Liu, Y. H. Ho, and K. A. Hua, "Privacy protected query processing with Road Network Embedding," in *Proc. of AINA*, 2011, pp. 481–487.

[6] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org," 2014.

[7] B. Yang, M. Kaul, and C. S. Jensen, "Using Incomplete Information for Complete Weight Annotation of Road Networks," *TKDE*, vol. 26, no. 5, pp. 1267–1279, 2014.

[8] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, "Short-term traffic forecasting: Where we are and where we're going," *Transportation Research Part C: Emerging Technologies*, vol. 43, pp. 3–19, 2014.

[9] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *Bulletin of TCDE*, vol. 40, no. 3, pp. 52–74, 2017.

[10] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online Learning of Social Representations," in *Proc. of SIGKDD*, 2014, pp. 701–710.

[11] A. Grover and J. Leskovec, "node2vec: Scalable Feature Learning for Networks," in *Proc. of SIGKDD*, 2016, pp. 855–864.

[12] L. F. R. Ribeiro, P. H. P. Saverese, and D. R. Figueiredo, "struc2vec: Learning Node Representations from Structural Identity," in *Proc. of SIGKDD*, 2017, pp. 385–394.

[13] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Proc. of NIPS*, 2017, pp. 1024–1034.

[14] X. Huang, J. Li, and X. Hu, "Label Informed Attributed Network Embedding," in *Proc. of WSDM*, 2017, pp. 731–739.

[15] C. Tu, H. Liu, Z. Liu, and M. Sun, "CANE: Context-Aware Network Embedding for Relation Modeling," in *Proc. of ACL*, vol. 1, 2017, pp. 1722–1731.

[16] L. Liao, X. He, H. Zhang, and T.-S. Chua, "Attributed Social Network Embedding," *TKDE*, 2018, in Press.

[17] M. Qu, J. Tang, J. Shang, X. Ren, M. Zhang, and J. Han, “An Attention-based Collaboration Framework for Multi-View Network Representation Learning,” in *Proc. of CIKM*, 2017, pp. 1767–1776.

[18] S. Pan, J. Wu, X. Zhu, C. Zhang, and Y. Wang, “Tri-Party Deep Network Representation,” in *Proc. of IJCAI*, 2016, pp. 1895–1901.

[19] L. Breiman, “Bagging predictors,” *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.

[20] L. van der Maaten, “Accelerating t-sne using tree-based algorithms,” vol. 15, pp. 3221–3245, 2014.

Paper B

Graph Convolutional Networks for Road Networks

Tobias Skovgaard Jepsen
Aalborg University

Christian S. Jensen
Aalborg University

Thomas Dyhre Nielsen
Aalborg University

This paper is published in the
Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,
pp. 460–463, 2019.

Abstract

The application of machine learning techniques in the setting of road networks holds the potential to facilitate many important transportation applications. Graph Convolutional Networks (GCNs) are neural networks that are capable of leveraging the structure of a network. However, many implicit assumptions of GCNs do not apply to road networks.

We introduce the Relational Fusion Network (RFN)*, a novel type of GCN designed specifically for road networks. In particular, we propose methods that substantially outperform state-of-the-art GCNs on two machine learning tasks in road networks. Furthermore, we show that state-of-the-art GCNs fail to effectively leverage road network structure on these tasks.*

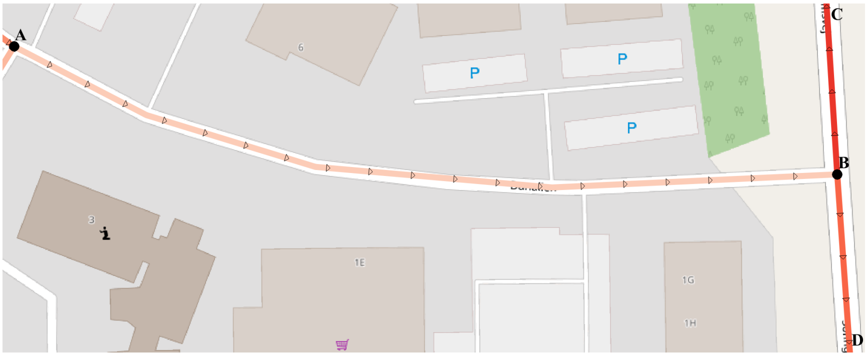


Fig. 1: Volatile homophily in a three-way intersection.

1 Introduction

Machine learning on road networks can facilitate important transportation applications such as traffic forecasting [1], speed limit annotation [2], and travel-time estimation. However, machine learning on road networks is difficult due to the low number of attributes, often with missing values, that typically are available [2]. This lack of attribute information can be alleviated by exploiting the network structure into the learning process [2]. To this end, we propose the *Relational Fusion Network (RFN)*, a type of Graph Convolutional Network (GCN) designed specifically for road networks.

GCNs are neural networks that operate directly on graph representations of networks. GCNs can in theory leverage road network structure by aggregating over a road segment's neighborhood when computing the segment's representation, e.g., computing the mean representations of its adjacent road segments. However, state-of-the-art GCNs are designed for node classification tasks in social, citation, and biological networks. Although GCNs have been highly successful at such tasks, machine learning tasks in road networks differ substantially.

First, many implicit assumptions in GCN proposals do not hold in the context of road networks. First, road networks are *edge-relational* and contain not only node and edge attributes, but also *between-edge attributes* that characterize the relationships between road segments (edges). For instance, the angle between two road segments is informative for travel time estimation since it influences the time it takes to move from one segment to the other.

Second, GCNs implicitly assume that the underlying network is homophilic meaning that adjacent road segments tend to be similar, and that changes in network characteristics, e.g., driving speeds, occur gradually. Although road networks exhibit homophily, the homophily is volatile in the sense that homophilic regions have sharp boundaries characterized by abrupt changes in, e.g., driving speeds. In the most extreme case, a region may consist of a single road segment, *in which case there is no homophily*. As an example, the three-way intersection to the right in Figure 1 exhibits volatile homophily. The two vertical road segments to the right and the road segments connected to the intersection to the form two regions that each is internally homophilic: the road segments within each region have similar driving speeds. The two regions are adjacent, but, a driver moving from one region to the other experiences an abrupt change in driving speed.

Contributions. We introduce the *Relational Fusion Network (RFN)*, a type of GCN designed specifically to address the shortcomings of state-of-the-art GCNs in the road network setting.[1] A novel relational fusion operator is at the core of a RFN. This graph convolutional operator aggregates over representations of relations instead of over representations of neighbors. To learn a representation of a relation (u, v), an RFN uses a fusion function that represents a relation (u, v) by fusing the representations, e.g., attributes, of road segments u and v and the attributes of their relation (u, v) that describe the nature of the relationship between u and v. This fusion mechanism allows an RFN to capture volatile homophily and makes it robust to aberrant neighbors in small neighborhoods.

RFNs are capable of leveraging node attributes, edge attributes, and *between-edge attributes* jointly during the learning process by considering both a *node view* and an *edge view*: two perspectives that capture the relationships between intersections and road segments, respectively. In comparison, state-of-the-art GCNs consider at most one of these perspectives and can leverage only one source of attributes. We evaluate the proposed RFN architecture on two road segment prediction tasks and find that the RFNs outperform state-of-the-art GCNs significantly on both tasks. Interestingly, our results suggest that an RFN can leverage neighborhood information in cases where state-of-the-art GCNs cannot.

The remainder of the paper is structured as follows. In Section 2, we give the necessary background on graph modeling of road networks and GCNs. In Section 3, we describe RFNs in detail. In Section 4, we report on empirical studies. Finally, we conclude in Section 5.

2 Preliminaries

Road Network Modeling We model a road network as an attributed, directed graph $G = (V, E, A^V, A^E, A^B)$, where V is the set of nodes and E is the set of edges. Each node $v \in V$ represents an intersection (or the end of

[1] Due to page limitation, we give only an introduction of our method in this paper. See [3] for a detailed description.

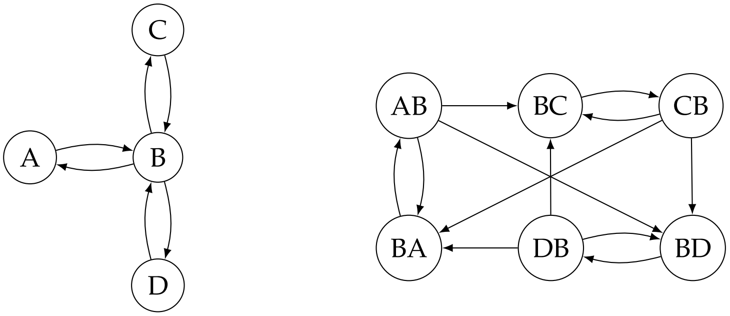


Fig. 2: The (left) primal and (right) dual graph representations of the three-way intersection to the right in Figure 1.

a road), and each edge $(u, v) \in E$ represents a road segment that enables traversal from u to v. Next, A^V and A^E maps intersections and road segments, respectively, to their attributes. In addition, A^B maps a pair of road segments $(u, v), (v, w) \in E$ to their between-segment attributes such as the angle between (u, v) and (v, w) based on their spatial representation. An example of a graph representation of the three-way intersection to the right in Figure 1 is shown to the left in Figure 2. Attribute information not shown.

Two intersections u and v in V are adjacent if $(u, v) \in E$ or $(v, u) \in E$. Similarly, two road segments (u_1, v_1) and (u_2, v_2) in E are adjacent if $v_1 = u_2$ or $v_2 = u_1$. The function $N \colon V \cup E \to 2^V \cup 2^E$ returns the neighborhood, i.e., the set of all adjacent intersections or road segments, of a road network element $g \in V \cup E$. The dual graph representation of G given by $G^D = (E, B)$, where $B = \big\{\big((u, v), (v, w)\big) \mid (u, v), (v, w) \in E\big\}$ is the set of *between-edges*. Thus, E and B are the node and edge sets, respectively, in the dual graph. An example of a dual graph can be seen to the right in Figure 2. For disambiguation, we refer to G as the primal graph representation.

Graph Convolutional Networks A GCN is a neural network that operates on graphs and consists of one or more graph convolutional layers. A graph convolutional network takes as input a graph $G = (V, E)$ and a numeric node feature matrix $\mathbf{X}^V \in \mathbb{R}^{|V| \times d_{in}}$, where each row corresponds to a d_{in}-dimensional vector representation of a node. Given these inputs, a GCN computes an output at a layer k s.t.

$$\mathbf{H}_v^{(V,k)} = \sigma(\textsc{Aggregate}^k(\{\mathbf{H}_n^{(V,k)} \mid n \in N(v)\})\mathbf{W}), \tag{B.1}$$

where σ is an activation function, and $\textsc{Aggregate} \colon 2^V \to \mathbb{R}^{d_{in}}$ is an aggregate function, e.g., a mean. As in $\mathbf{X}^V$, each row in $\mathbf{H}^{(V,k)}$ is a vector representation of a node. In some cases, $\mathbf{X}^V$ is linearly transformed using matrix multiplication with a weight matrix $\mathbf{W}$ before aggregation [4], while in other cases, weight multiplication is done after aggregation [5, 6], as in Equation B.1.

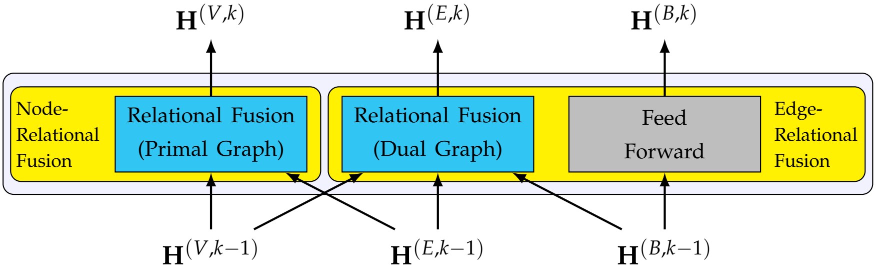


Fig. 3: Relational Fusion Layer.

3 Relational Fusion Networks

Relational Fusion Networks (RFNs) aim to address the shortcomings of state-of-the-art GCNs in the context of machine learning on road networks. We now proceed to give a brief introduction of our method. A more detailed description may be found in the full paper [3].

3.1 Overview

The basic premise of the RFN is to learn representations based on two distinct, but interdependent, views: the node-relational and edge-relational views. An RFN consists of K relational fusion layers, where $K \geq 1$. We illustrate a single relational fusion layer in Figure 3.

Each layer k takes as input the learned node, edge, and between-edge representations from layer $k-1$, denoted by $\mathbf{H}^{(V,k-1)}$, $\mathbf{H}^{(E,k-1)}$, and $\mathbf{H}^{(B,k-1)}$, respectively. The first layer takes as input the feature matrices $\mathbf{X}^V \in \mathbb{R}^{|V| \times d^V}$, $\mathbf{X}^E \in \mathbb{R}^{|E| \times d^E}$, and $\mathbf{X}^B \in \mathbb{R}^{|B| \times d^B}$ that numerically encode the node, edge, and between-edge attributes, respectively. Then *node-relational fusion* and *edge-relational fusion* are performed to learn new node and edge representations $\mathbf{H}^{(V,k)}$ and $\mathbf{H}^{(E,k)}$ from the node- and edge-relational views, respectively.

Using node-relational fusion, we seek to learn representations of nodes, i.e., intersections, based on their node attributes and the relationships between nodes indicated by the edges E in the primal graph $G^P = (V, E)$ and described by their edge attributes. Similarly, we seek to learn representations of edges, i.e., road segments, using edge-relational fusion, based on their edge attributes and the relationships between edges indicated by the between-edges B in the dual graph $G^D = (E, B)$. The relationship between two adjacent roads (u, v) and (v, w) is described by the attributes of the between-edge connecting them in the dual graph, including the angle between them, but also the attributes of the node v that connects them. These node and edge views are interdependent and can be exploited by RFNs to leverage node, edge, and between-edge attributes simultaneously.

As illustrated in Figure 3, an RFN captures the interdependence between the node and edge views by using the node and edge representations from the previous layer $k-1$ as input to the node-relational and edge-relational fusion in layer k In addition, each layer applies a regular feed-forward neural network to the between-edge presentations $\mathbf{H}^{(B,k-1)}$ to learn new between-edge representations $\mathbf{H}^{(B,k)}$.

3.2 Relational Fusion

We present the pseudocode for relational fusion at the kth layer in Algorithm 2. The operator takes as input a graph $G' = (V', E')$, that is either the primal or dual graph representation of a road network, along with appropriate feature matrices $\mathbf{H}^{(V',k-1)}$ and $\mathbf{H}^{(E',k-1)}$ that describe nodes and edges in G'. Then, a new representation is computed for each element $v' \in V'$ by first computing relational representations. Given an element v', each relation $(v', n') \in N(v')$ that v' participates in, is converted to a *relational representation*. To be explicit, $G' = G^P = (V, E)$, $\mathbf{H}^{(V',k-1)} = \mathbf{H}^{(V,k-1)}$, and $\mathbf{H}^{(E',k-1)} = \mathbf{H}^{(E,k-1)}$ in the case of node-relational fusion. In the case of edge-relational fusion, $G' = G^D = (E, B)$, $\mathbf{H}^{(V',k-1)} = \mathbf{H}^{(E,k-1)}$, and $\mathbf{H}^{(E',k-1)}$ combines node and between-edge features, e.g., s.t. the representation of a between-edge $((u,v),(v,w)) \in B$ is $\mathbf{H}^{(E',k-1)}_{((u,v),(v,w))} = \mathbf{H}^{(B,k-1)}_{((u,v),(v,w))} \oplus \mathbf{H}^{(V,k-1)}_{v}$, where $\oplus$ denotes vector concatenation.

Algorithm 2 The Relational Fusion Operator

```
function RelationalFusion^k($G' = (V', E')$, $\mathbf{H}^{(V',k-1)}$, $\mathbf{H}^{(E',k-1)}$)
    let $\mathbf{H}^{(V',k)}$ be an arbitrary $|V'| \times d^{F^k}$ real feature matrix.
    for all $v' \in V'$ do
        $F_{v'} \leftarrow \{$
                Fuse^k($\mathbf{H}^{(V',k-1)}_{v'}, \mathbf{H}^{(E',k-1)}_{(v',n')}, \mathbf{H}^{(V',k-1)}_{n'}$) $\mid n' \in N(v')\}$
        $\mathbf{H}^{(V',k)}_{v'} \leftarrow$ Aggregate^k($F_{v'}$)
        $\mathbf{H}^{(V',k)}_{v'} \leftarrow$ Normalize^k($\mathbf{H}^{(V',k)}_{v'}$)
    return $\mathbf{H}^{(V',k)}$
```

In Algorithm 2, the relational representations at layer k are computed by a fusion function Fuse^k. For each relation, Fuse^k takes as input representations of the source v' and target n' of the relation, $\mathbf{H}^{(V',k-1)}_{v'}$ and $\mathbf{H}^{(V',k-1)}_{n'}$, respectively, along with a representation $\mathbf{H}^{(E',k-1)}_{(v',n')}$ describing their relation, and then it fuses them. The resulting relational representations are subsequently fed to an Aggregate^k function, that aggregates them into a single representation of

v'. Finally, the representation of v' may optionally be normalized by invoking the $\textsc{Normalize}^k$ function., e.g., using L_2 normalization [6]. This latter step is particularly important if the relational aggregate has different scales across elements with different neighborhood sizes.

The relational fusion operator is compatible with many existing aggregators from the GCN literature, e.g., a mean aggregator [6]. We use a single-layer perceptron as the fusion function, i.e.,

$$\begin{aligned}\textsc{Fuse}^k(\mathbf{H}_{v'}^{(V',k-1)}, \mathbf{H}_{n'}^{(V',k-1)}, \mathbf{H}_{(v',n')}^{(E',k-1)}) = \\ \sigma\big((\mathbf{H}_{v'}^{(V',k-1)} \oplus \mathbf{H}_{n'}^{(V',k-1)} \oplus \mathbf{H}_{(v',n')}^{(E',k-1)})\mathbf{W}^R + \mathbf{b}\big),\end{aligned}$$

where σ is an activation function, $\oplus$ denotes row-wise vector concatenation, $\mathbf{W}^R$ is a weight matrix, and $\mathbf{b}$ is a bias term. We explore aggregator and fusion function designs in the full paper [3].

4 Experimental Evaluation

To investigate the generality of our method, we evaluate it on two tasks using the road network of the Danish municipality of Aalborg: driving speed estimation and speed limit classification. These tasks represent a regression task and a classification task, respectively.

Many details of the experiments have been omitted due to the page limitation. We refer to the full paper [3] for further information. Our RFN implementation is available online[2].

4.1 Data Set

We extract the spatial representation of the Danish municipality of Aalborg from OSM [7], and convert it to its primal and dual graph representations as described in Section 2. We combine the OSM data with a zone map from the Danish Business Authority[3], and we derive 3 node features, 16 edge features, and 2 between-edge features from this dataset.

For the driving speed estimation task, we use a dataset of 8 675 599 observed driving speeds, each matched to a road segment, that stem from a set of vehicle trajectories [8]. For the speed limit classification task, we use 19 510 speed limits collected from the OSM data and additional speed limits are collected from the municipality of Aalborg. This dataset is highly imbalanced. Finally, we split speed limits and driving speeds into training, validation, and test sets.

[2] `https://github.com/TobiasSkovgaardJepsen/relational-fusion-networks`

[3] `https://danishbusinessauthority.dk/plansystemdk`

4.2 Experimental Setup

We compare four algorithms in our experiments:

- *MLP*: A regular multi-layer perceptron that performs predictions independent of adjacent road segments by using only the edge features as input.
- *GraphSAGE*: The Max-Pooling variant of GraphSAGE, which achieved the best results in the authors' experiments [6].
- *GAT*: The graph attention network by Veličković et al. [4].
- *RFN*: An RFN using a mean aggregator [6].

The GraphSAGE and GAT models are run on the dual graph representations of the road network s.t. they learn edge representations directly. All models are two-layer models and use the ELU [9] activation function, with the exception that the ReLU [10] activation function is used in the GraphSAGE pooling operation. We select layer sizes, learning rates, and GAT-specific hyperparameters by evaluating different hyperparameter configurations on the validation sets in a grid search and selecting the best-performing configuration.

All algorithms are implemented using the MXNet[4] deep learning library.

Model Training and Evaluation We initialize the weights of all models using Xavier initialization [11] and train the models using the ADAM optimizer [12] in batches of 256 segments. In preliminary experiments, we observed that all models converged within 20 and 30 epochs for driving speed estimation and speed limit classification, respectively. We therefore use these values for training. For speed limit classification, we use random oversampling on the training set to handle the class imbalance in the dataset and use early stopping to regularize the model.

To train the models, we minimize a per-segment mean squared loss and the binary cross entropy loss for driving speed estimation and speed limit classification, respectively. To evaluate the models, we use a per-segment mean absolute error for driving speed estimation and the F_1 macro score for speed limit classification.

4.3 Results

We report the mean performance and standard deviations of each algorithm across ten runs in Table 1. Note that when reading Table 1, low values and high values are desirable for driving speed estimation and speed limit classification, respectively.

[4] `https://mxnet.incubator.apache.org/`

Table 1: Algorithm performance on Driving Speed Estimation (DSE) and Speed Limit Classification (SLC).

Algorithm	*DSE*	*SLC*
MLP	10.160 ± 0.119	0.443 ± 0.027
GraphSAGE	8.960 ± 0.115	0.432 ± 0.014
GAT	9.548 ± 0.151	0.442 ± 0.018
RFN	$\mathbf{7.685 \pm 0.189}$	$\mathbf{0.500 \pm 0.011}$

As can be seen, our proposed RFN outperforms all baselines on both driving speed estimation and speed limit classification. RFN outperforms the state-of-the-art graph convolutional approaches, i.e., GraphSAGE and GAT, by 17% and 24%, respectively, on the driving speed estimation task. On the speed limit classification task, the best RFN outperforms GraphSAGE and GAT by 16% and 13%, respectively. The more sophisticated aggregation and fusion functions that we present in the full version of the paper substantially improve these results s.t. the best RFN variant outperforms GraphSAGE and GAT by 32–40% and 21–24%, respectively [3].

Interestingly, the MLP outperforms the GraphSAGE and GAT (but not the RFN) models on speed limit classification without using the network structure. This suggests that RFNs can leverage road network structure in cases where GraphSAGE and GAT cannot.

5 Conclusion

We report on a study of GCNs from the perspective of machine learning on road networks. We argue that many built-in assumptions of existing proposals do not apply in the road network setting, in particular the assumption of smooth homophily in the network. In addition, state-of-the-art GCNs can leverage only one source of attribute information, whereas we identify three sources of attribute information in road networks: node, edge, and between-edge attributes. To address these short-comings, we propose the *Relational Fusion Network (RFN)*, a novel type of GCN for road networks.

We compare the RFN against state-of-the-art GCN algorithms on two machine learning tasks in road networks. We find that the proposed RFN outperforms the GCN baselines significantly on these tasks. Although not presented here, we also investigate alternative aggregation and fusion functions that yield even higher predictive performance [3].

In future work, it is of interest to investigate to which extent RFNs are capable of transferring knowledge from, e.g., one Danish municipality to the rest of Denmark, given that the inductive nature of our algorithm allows RFNs trained on one road network to be used for prediction on another. If the results are positive, it would suggest that RFNs can learn traffic dynamics that generalize to unseen regions of the network. This may make it easier to train RFNs with less data, but also give more confidence in predictions in regions that are labeled sparsely with speed limits. In addition, RFNs do not incorporate temporal aspects, although many road networks tasks are time-dependent. For instance, this applies to driving speed estimation, for which reason we explicitly excluded driving speeds during peak-hours from our experiments. Extending RFNs to learn temporal road network dynamics, e.g., through time-dependent fusion functions that accept temporal inputs, is an important future direction.

6 Acknowledgments

The research presented in this paper is supported in part by the DiCyPS project and by grants from the Obel Family Foundation and the Villum Fonden.

References

[1] B. Yu, H. Yin, and Z. Zhu, "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting," in *Proc. of IJCAI*, 2017, pp. 3634–3640.

[2] T. S. Jepsen, C. S. Jensen, T. D. Nielsen, and K. Torp, "On Network Embedding for Machine Learning on Road Networks: A Case Study on the Danish Road Network," in *Proc. of Big Data*, 2018, pp. 3422–3431.

[3] T. S. Jepsen, C. S. Jensen, and T. D. Nielsen, "Graph Convolutional Networks for Road Networks," *arXiv e-prints*, 2019.

[4] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," in *Proc. of ICLR*, 2018, p. 12 pp.

[5] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *Proc. of ICLR*, 2017, p. 14 pp.

[6] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Proc. of NIPS*, 2017, pp. 1024–1034.

[7] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org," 2014.

[8] O. Andersen, B. B. Krogh, and K. Torp, "An Open-source Based ITS Platform," in *Proc. of MDM*, vol. 2, 2013, pp. 27–32.

[9] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs) ,” in *Proc. of ICLR*, 2016, p. 14 pp.

[10] X. Glorot, A. Bordes, and Y. Bengio, “Deep Sparse Rectifier Neural Networks,” in *Proc. of AISTATS*, 2011, pp. 315–323.

[11] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks,” in *Proc. of AISTATS*, 2010, pp. 249–256.

[12] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” in *Proc. of ICLR*, 2015, p. 15 pp.

Paper C

Relational Fusion Networks: Graph Convolutional Networks for Road Networks

Tobias Skovgaard Jepsen
Aalborg University

Christian S. Jensen
Aalborg University

Thomas Dyhre Nielsen
Aalborg University

This paper is awaiting publication in
IEEE Transactions on Intelligent Transportation Systems,
2020. Available in early online access.

Abstract

The application of machine learning techniques in the setting of road networks holds the potential to facilitate many important intelligent transportation applications. Graph Convolutional Networks (GCNs) are neural networks that are capable of leveraging the structure of a network. However, many implicit assumptions of GCNs do not apply to road networks.

We introduce the Relational Fusion Network (RFN)*, a novel type of GCN designed specifically for road networks. In particular, we propose methods that outperform state-of-the-art GCNs by* 21%-40% *on two machine learning tasks in road networks. Furthermore, we show that state-of-the-art GCNs may fail to effectively leverage road network structure and may not generalize well to other road networks.*

Tobias Skovgaard Jepsen, Christian S. Jensen, and Thomas Dyhre Nielsen, 'Relational Fusion Networks: Graph Convolutional Networks for Road Networks.' *IEEE Transactions on Intelligent Transportation Systems*, 2020. DOI: 10.1109/TITS.2020.3011799.

The layout has been revised.

1 Introduction

Machine learning on road networks can facilitate important intelligent transportation applications such as traffic flow prediction [1, 2], traffic speed forecasting [3, 4], speed limit annotation [5], and travel time estimation [6, 7]. However, machine learning on road networks is difficult due to the low number of attributes, often with missing values, that typically are available [5]. This lack of attribute information can be alleviated by exploiting the network structure into the learning process [5]. To this end, we propose the *Relational Fusion Network (RFN)*, a type of Graph Convolutional Network (GCN) designed for machine learning on road networks.

GCNs are a type of neural network that operates on graph representations of networks. GCNs can in theory leverage the road network structure by aggregating over a road segment's neighborhood when computing the segment's representation, e.g., computing the mean representations of its adjacent road segments.

State-of-the-art GCNs are designed to target node classification tasks in social, citation, and biological networks [8–13]. Such networks differ substantially from road networks in terms of the attribute information available and other network characteristics. As a result, many implicit assumptions in GCN proposals do not hold.

First, road networks are *edge-relational* and contain not only node and edge attributes, but also *between-edge attributes* that characterize the relationship between road segments, i.e., the edges in a road network. For instance, the angle between two road segments is informative for travel time estimation since it influences the time it takes to move from one segment to the other.

Second, compared to social, citation, and biological networks, road networks are low-density: road segments have few adjacent road segments. For instance, the Danish road network has a mean node degree of 2.2 [5] compared to the mean node degrees 9.15 and 492 of a citation and a social networks, respectively [11]. The small neighborhoods in road networks make neighborhood aggregation in GCNs sensitive to aberrant neighbors, which may contribute noise.

Third, GCNs implicitly assume that the underlying network exhibits homophily meaning that adjacent road segments tend to be similar, and that changes in network characteristics, e.g., driving speeds, occur gradually. Although road networks exhibit homophily, the homophily is volatile in the sense that regions can be highly homophilic, but have sharp boundaries characterized by abrupt changes in, e.g., driving speeds. This might for instance be caused by a change in speed limit such as when exiting a motorway. In the most extreme case, a region may consist of a single road segment, *in which case there is no homophily*.

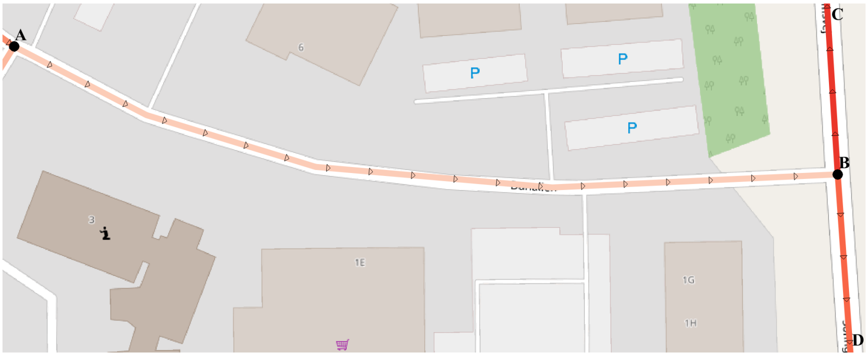


Fig. 1: Two three-way intersections in Denmark. We illustrate the observed driving speeds for one driving direction per segment and color them accordingly: the darker, the faster the speed. Black dots mark intersections and triangles indicate driving directions.

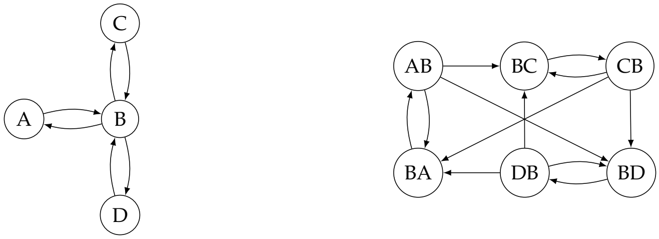
As an example, the three-way intersection to the right in Figure 1 exhibits of volatile homophily. The two vertical road segments to the right in the figure and the road segments to the left in the figure form two internally homophilic regions: within each region, the road segments exhibit similar driving speeds. The two regions are adjacent in the network, but a driver moving from one region to the other experiences an abrupt change in driving speed.

We suggest that new GCN architectures are needed for high-performance machine learning on road networks based on our observations regarding current state-of-the-art GCNs. Addressing the challenge of volatile homophily is of particular importance. We therefore propose the Relational Fusion Network (RFN), a novel GCN architecture designed to be generally applicable to machine learning tasks on road networks.

Unlike GCNs, RFNs take into account the inherent properties of road networks, i.e., that they are edge-relational, low-density, and exhibit volatile homophily. Specifically, RFNs (i) explicitly incorporate the relationships between edges during aggregation, (ii) use an attention mechanism to exclude noise-contributing neighbors during aggregation, and (iii) use a relational fusion operator that allows an RFN to only conditionally rely on the homophily assumption when performing neighborhood aggregation.

We experimentally evaluate different RFN variants on two example machine learning tasks on road networks: driving speed estimation and speed limit classification. Our experiments show that the best RFN variants outperform state-of-the-art GCNs by 32–40% and 21–24% on driving speed estimation and speed limit classification, respectively.

In our experiments, we demonstrate that state-of-the-art GCNs fail to leverage road network structure on the speed limit classification task where



(a) Primal Graph.

(b) Dual Graph.

Fig. 2: The (a) primal and (b) dual graph representations of the three-way intersection to the right in Figure 1.

RFNs do not. Finally, we show that the knowledge learned by an RFN can generalize well to an entirely unseen part of the road network. This feature is particular important in cases where there is a large spatial imbalance, e.g., when using crowd-sourced data or vehicle trajectory data. In such data, data points tend to be concentrated in areas with high population density such as in major cities.

The remainder of the paper is structured as follows. In Section 5, we review related work. In Section 2, we give the necessary background on graph modeling of road networks and GCNs. In Section 3, we describe RFNs in detail. In Section 4, we describe our experiments and present our results. Finally, we conclude in Section 6.

2 Preliminaries

We now cover the necessary background in modeling road networks as graphs and GCNs.

2.1 Modeling Road Networks

We model a road network as an attributed directed graph $G = (V, E, A^V, A^E, A^B)$ where V is the set of nodes and E is the set of edges. Each node $v \in V$ represents an intersection, and each edge $(u, v) \in E$ represents a road segment that enables traversal from u to v. Next, A^V and A^E maps intersections and road segments, respectively, to their attributes. In addition, A^B maps a pair of road segments $(u, v), (v, w) \in E$ to their between-segment attributes such as the angle between (u, v) and (v, w) based on their spatial representation. An example of a graph representation of the three-way intersection to the right in figure Figure 1 is shown in Figure 2a.

Two intersections u and v in V are adjacent if there exists a road segment $(u, v) \in E$ or $(v, u) \in E$. Similarly, two road segments (u_1, v_1) and (u_2, v_2) in E are adjacent if $v_1 = u_2$ or $v_2 = u_1$. The function $N\colon V \cup E \to 2^V \cup 2^E$ returns the neighborhood, i.e., the set of all adjacent intersections or road segments, of a road network element $g \in V \cup E$. The dual graph representation of G is then $G^D = (E, B)$ where $B = \big\{\big((u,v),(v,w)\big) \mid (u,v),(v,w) \in E\big\}$ is the set of *between-edges*. Thus, E and B serve as the node and edge sets of the dual graph, respectively. For disambiguation, we refer to G as the primal graph representation.

2.2 Graph Convolutional Networks

A GCN is a neural network that operates on graphs and consists of one or more graph convolutional layers. A graph convolutional network takes as input a graph $G = (V, E)$ and a numeric node feature matrix $\mathbf{X}^V \in \mathbb{R}^{|V| \times d_{in}}$, where each row corresponds to a d_{in}-dimensional vector representation of a node. Given these inputs, a GCN computes an output at a layer k s.t.

$$\mathbf{H}_v^{(V,k)} = \sigma(\textsc{Aggregate}^k(v)\mathbf{W}^k), \tag{C.1}$$

where σ is an activation function, $\textsc{Aggregate}\colon 2^V \to \mathbb{R}^{d_{in}}$ is a neighborhood aggregation function, and $\mathbf{W}^k \in \mathbb{R}^{d_{in} \times d_o}$ is a learned weight matrix. Similarly to $\mathbf{X}^V$, each row in $\mathbf{H}^{(V,k)}$ is a vector representation of a node. Note that in some cases $\mathbf{X}^V$ is linearly transformed using matrix multiplication with a weight matrix $\mathbf{W}^k$ before aggregation [12] while in other cases, weight multiplication is done after aggregation [10, 11], as in Equation C.1.

The $\textsc{Aggregate}$ function in Equation C.1 derives a new representation of a node v by aggregating over the representations of its neighbors. While the aggregate function is what distinguishes GCN architectures from each other, many can be expressed as a weighted sum [10–12]:

$$\textsc{Aggregate}^k(v) = \sum_{n \in N(v)} a_{(v,n)} \mathbf{H}_n^{(V,k-1)}, \tag{C.2}$$

where $\mathbf{H}^{(V,0)} = \mathbf{X}^V$, and $a_{(v,n)}$ is the aggregation weight for neighbor n of node v. For instance, $a_{(v,n)} = |N(v)|^{-1}$ in the mean aggregator of GraphSAGE [11].

3 Proposed Method

The *Relational Fusion Network (RFN)* aims to address the shortcomings of state-of-the-art GCNs in the context of machine learning on road networks. The basic premise is to learn representations based on two distinct, but interdependent views: the node-relational and edge-relational views.

3.1 Node-Relational and Edge-Relational Views

In the node-relational view, we seek to learn representations of nodes, i.e., intersections, based on their node attributes and the relationships between nodes indicated by the edges E in the primal graph representation of a road network $G^P = (V, E)$ and described by their edge attributes. Similarly, we seek to learn representations of edges, i.e., road segments, in the edge-relational view, based on their edge attributes and the relationships between edges indicated by the between-edges B in the dual graph representation of a road network $G^D = (E, B)$. The relationship between two adjacent roads (u, v) and (v, w) is described by the attributes of the between-edge connecting them in the dual graph, including the angle between them, but also the attributes of the node v that connects them.

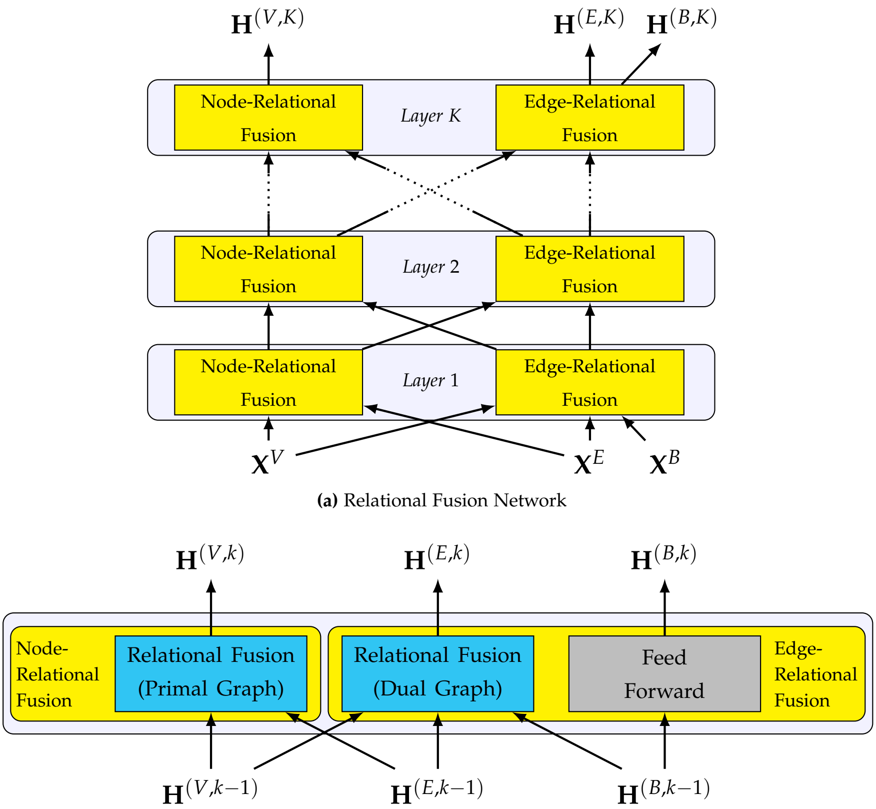
The node-relational and edge-relational views are complementary. The representation of a node in the node-relational view is dependent on the representation of the edges to its neighbors. Similarly, the representation of an edge in the edge-relational view is dependent on the representation of the nodes that it shares with its neighboring edges. Finally, the representation of an edge is also dependent on the representation of the between-edge connecting them in the dual graph. RFNs can exploit these two complementary views to leverage node, edge, and between-edge attributes simultaneously.

3.2 Method Overview

Figure 3a gives an overview of our method. As shown, an RFN consists of K relational fusion layers, where $K \geq 1$. It takes as input feature matrices $\mathbf{X}^V \in \mathbb{R}^{|V| \times d_V}$, $\mathbf{X}^E \in \mathbb{R}^{|E| \times d_E}$, and $\mathbf{X}^B \in \mathbb{R}^{|B| \times d_B}$ that numerically encode the node, edge, and between-edge attributes, respectively. These inputs are propagated through each layer. Each of these relational fusion layers, performs *node-relational fusion* and *edge-relational fusion* to learn representations from the node-relational and edge-relational views, respectively.

Node-relational fusion is carried out by performing relational fusion on the primal graph. Relational fusion is a novel graph convolutional operator that we describe in detail in Section 3.3. In brief, relational fusion computes relational representations for, e.g., each relation (v, n) of a node v in the primal graph. These relational representations are a fusion of the node representations of v and n, but also the edge representation of (v, n). Finally, the relational representations are aggregated into a new representation of v. Relational fusion differs from regular graph convolution by replacing the aggregate over neighbor representations with an aggregate over relational representations.

Edge-relational fusion is performed similarly to node-relational fusion but is applied on the dual graph representation. Recall, that in the edge-relational



(a) Relational Fusion Network

(b) Relational Fusion Layer

Fig. 3: Overview of our method showing (a) a K-layered relational fusion network and (b) a relational fusion layer.

view, the relation between two edges (u, v) and (v, w) is in part described by their between-edge attributes, but also by the node attributes of v that describes the characteristics of the intersection between them. Thus, as illustrated in Figure 3b, relational fusion on the dual graph requires both node and between-edge information to compute relational aggregates.

The interdependence between the node-relational and edge-relational views is captured by using the node and edge representations from the previous layer $k - 1$ as input to node-relational and edge-relational fusion in the next layer k as illustrated by Figure 3. Therefore each layer also applies regular feed-forward propagation on the between-edge representations to learn more abstract between-edge representations as input to the next layer. Unlike node-relational and edge-relational fusion, this feed-forward propagation does not consider neighborhood information.

Figure 3b gives a more detailed view of a relational fusion layer. Each layer k takes as input the learned node, edge, and between-edge representations from layer $k-1$, denoted by $\mathbf{H}^{(V,k-1)}$, $\mathbf{H}^{(E,k-1)}$, and $\mathbf{H}^{(B,k-1)}$, respectively. Then node-relational and edge-relational fusion are performed to output new node, edge, and between-edge representations $\mathbf{H}^{(V,k)}$, $\mathbf{H}^{(E,k)}$, and $\mathbf{H}^{(B,k)}$.

3.3 Relational Fusion

We present the pseudocode for the relational fusion operator at the kth layer in Algorithm 3. For clarity of notation, we write the notation from the perspective of a primal graph, but it can be applied to any graph, including a dual graph. The operator takes as input a graph $G = (V, E)$, that is either the primal or dual graph representation of a road network, along with appropriate feature matrices $\mathbf{H}^{(V,k-1)}$ and $\mathbf{H}^{(E,k-1)}$ to describe nodes and edges in G. Then, a new representation is computed for each node $v \in V$ by first computing relational representations at line 4. Given a node v, each relation $(v, n) \in N(v)$ that v participates in, is converted to a *relational representation*.

Algorithm 3 The Node-Relational Fusion Operator

```
1: function RelationalFusion$^k$($G = (V, E)$, $\mathbf{H}^{(V,k-1)}$, $\mathbf{H}^{(E,k-1)}$)
2:     let $\mathbf{H}^{(V,k)}$ be an arbitrary $|V| \times d_{(F,k)}$ real feature matrix.
3:     for all $v \in V$ do
4:         $F_v \leftarrow \{$
               Fuse$^k$($\mathbf{H}_v^{(V,k-1)}$, $\mathbf{H}_{(v,n)}^{(E,k-1)}$, $\mathbf{H}_n^{(V,k-1)}$) $\mid n \in N(v)\}$
5:         $\mathbf{H}_v^{(V,k)} \leftarrow$ Aggregate$^k$($F_v$)
6:         $\mathbf{H}_v^{(V,k)} \leftarrow$ Normalize$^k$($\mathbf{H}_v^{(V,k)}$)
7:     return $\mathbf{H}^{(V,k)}$
```

In Algorithm 3, the relational representations at layer k are computed by a fusion function Fuse^k that outputs $d_{(F,k)}$-dimensional relational representations. For each relation (v, n), Fuse^k takes as input representations of the source v and target n of the relation, $\mathbf{H}_v^{(V,k-1)}$ and $\mathbf{H}_n^{(V,k-1)}$, respectively, along with a representation $\mathbf{H}_{(v,n)}^{(E,k-1)}$ describing their relation, and fuses them into a relational representation of dimensionality $d_{(F,k)}$. The relational representations are subsequently fed to a Aggregate^k function, that aggregates them into a single d_o-dimensional representation of v' of dimensionality. Finally, the representation of v may optionally be normalized by invoking the Normalize^k function, e.g., using L_2 normalization [11]. The normalization step is particularly important if the relational aggregate has different scales across elements with different neighborhood sizes, e.g., if the aggregate is a sum.

Next, we discuss different choices for the fuse functions Fuse^k and the relational aggregators $\textsc{Aggregate}^k$.

3.4 Fusion Functions

The fusion function is responsible for extracting useful information from each relation, thereby allowing an RFN to create sharp boundaries at the edges of homophilic regions. The fusion function thus plays an important role w.r.t. capturing volatile homophily. We propose two fusion functions: AdditiveFuse and InteractionalFuse.

AdditiveFuse takes as input source, target, and relation representations $\mathbf{H}_v^{(V,k-1)}$, $\mathbf{H}_n^{(V,k-1)}$, and $\mathbf{H}_{(v,n)}^{(E,k-1)}$, and transforms these sources:

$$\textsc{AdditiveFuse}^k(\mathbf{H}_v^{(V,k-1)}, \mathbf{H}_n^{(V,k-1)}, \mathbf{H}_{(v,n)}^{(E,k-1)}) = \sigma\big(\mathbf{H}_{(v,n)}^{(R,k-1)}\mathbf{W}^R + \mathbf{b}\big), \quad \text{(C.3)}$$

where the relational feature matrix $\mathbf{H}_{(v,n)}^{(R,k-1)} \in \mathbb{R}^{d_R}$ is the vector concatenation of $\mathbf{H}_v^{(V,k-1)}$, $\mathbf{H}_n^{(V,k-1)}$, and $\mathbf{H}_{(v,n)}^{(E,k-1)}$. Next, σ is an activation function, $\mathbf{W}^R \in \mathbb{R}^{d_R \times d_o}$ is a weight matrix, $\mathbf{b} \in \mathbb{R}^{1 \times d_o}$ is a bias term, and d_o is the output dimensionality of the fusion function (and its parent relational fusion operator). Equation C.3 is additive in the sense that it is equivalent to transforming the source, target, and relational representations with independent weight matrices and summing them before applying activation function σ.

AdditiveFuse summarizes the relationship between v and n, but does not explicitly model interactions between representations. Hence, we propose an interactional fusion function:

$$\textsc{InteractionalFuse}^k(\mathbf{H}_v^{(V,k-1)}, \mathbf{H}_n^{(V,k-1)}, \mathbf{H}_{(v,n)}^{(E,k-1)}) = \sigma\big((\mathbf{H}^{(R,k-1)}\mathbf{W}^I \odot \mathbf{H}^{(R,k-1)})\mathbf{W}^R\big) + \mathbf{b}, \quad \text{(C.4)}$$

where $\mathbf{W}^I \in \mathbb{R}^{d_R \times d_R}$ is a trainable interaction weight matrix, $\odot$ denotes element-wise multiplication, and $\mathbf{b} \in \mathbb{R}^{1 \times d_o}$ is a bias term. Notice that InteractionalFuse has a quadratic form. When computing the term $\mathbf{H}^{(I,k)} = (\mathbf{H}^{(R,k-1)}\mathbf{W}^I) \odot \mathbf{H}^{(R,k-1)}$, the ith ($1 \leq i \leq d_R$) value of vector $\mathbf{H}^{(I,k)}$ is

$$\mathbf{H}_i^{(I,k)} = \sum_{j=1}^{d_R} \mathbf{W}_{i,j}^I \mathbf{H}_i^{(R,k-1)} \mathbf{H}_j^{(R,k-1)}. \quad \text{(C.5)}$$

In other words, $\mathbf{H}_i^{(I,k)}$ is the weighted sum of all interactions between the ith feature and all other features of the relational representation $\mathbf{H}^{(R,k-1)}$. Modeling these interaction terms explicitly enables InteractionalFuse to capture and weigh interactions at a much finer granularity than AdditiveFuse.

InteractionalFuse offers improved modeling capacity over AdditiveFuse to better address the challenge of volatile homophily, but at the cost of an increase in number of parameters that is quadratic in the number of input features.

3.5 Aggregation Functions

Different Aggregate functions have been proposed in the literature, and many are directly compatible with the relational fusion layer. For instance, mean, and max/mean pooling aggregators [11].

Recently, aggregators based on attention mechanisms from the domain of natural language processing have appeared [12]. Such graph attention mechanisms allow a GCN to filter out irrelevant or aberrant neighbors by weighing the contribution of each neighbor to the neighborhood aggregate. This filtering property is highly desirable for road networks where even a single aberrant neighbor can contribute significant noise to an aggregate due to the low density of road networks. In addition, it may help the network distinguish adjacent regions from each other and thereby improve predictive performance in the face of volatile homophily.

Current graph attention mechanisms rely on a common transformation of each neighbor thus rendering them incompatible with RFNs, which rely on the context-dependent neighbor transformations performed by the Fuse function at each relational fusion layer. We therefore propose an attentional aggregator that is compatible with our proposed RFNs.

Attentional Aggregator The attentional aggregator we propose computes a weighted mean over the relational representations. Formally, the attentional aggregator computes the Aggregate in Algorithm 3 as

$$\textsc{Aggregate}(F_v) = \sum_{\mathbf{f}_n \in F_v} A(v,n)\mathbf{f}_n, \tag{C.6}$$

where $F_v = \{\mathbf{f}_n \mid n \in N(v)\}$ is the set of fused relational representations of node v's relations to each of its neighbors $n \in N(v)$ computed at line 4 in Algorithm 3. Furthermore, A is an attention function, and $A(v,n)$ is the *attention weight* that determines the contribution of each neighbor n to the neighborhood aggregate of node v.

An attention weight $A(v,n)$ in Equation C.6 depends on the relationship between a node v and its neighbor n in the input graph G to Algorithm 3. This relationship is described by the relational feature matrix $\mathbf{H}^{(R,k-1)}_{(v,n)} \in \mathbb{R}^{d_R}$

and the attention weight is computed as

$$A(v,n) = \frac{\exp\left(C(\mathbf{H}^{(R,k-1)}_{(v,n)})\right)}{\sum_{m\in N(v)} \exp\left(C(\mathbf{H}^{(R,k-1)}_{(v,m)})\right)}, \tag{C.7}$$

where $C\colon \mathbb{R}^{d_R} \to \mathbb{R}$ is an *attention coefficient* function

$$C(\mathbf{H}^{(R,k-1)}_{(v,n)}) = \sigma(\mathbf{H}^{(R,k-1)}_{(v,n)}\mathbf{W}^{C}), \tag{C.8}$$

where $\mathbf{W}^{\mathbf{C}} \in \mathbb{R}^{d_R}$ is a weight matrix and σ is an activation function. This corresponds to computing the softmax over the attention coefficients of all neighbors n of v. All attention weights sum to one, i.e., $\sum_{n\in N(v)} A(v,n) = 1$, thus making Equation C.6 a weighted mean. In other words, each neighbor's contribution to the mean is regulated by an attention weight, thus allowing an RFN to reduce the influence of (or completely ignore) aberrant neighbors that would otherwise contribute significant noise to the neighborhood aggregate.

3.6 Forward Propagation

Algorithm 4 Forward Propagation Algorithm

```
function ForwardPropagation($\mathbf{X}^V$, $\mathbf{X}^E$, $\mathbf{X}^B$)
    let $\mathbf{H}^{(V,0)} = \mathbf{X}^V$, $\mathbf{H}^{(E,0)} = \mathbf{X}^E$, and $\mathbf{H}^{(B,0)} = \mathbf{X}^B$
    for $k = 1$ to $K$ do
        $\mathbf{H}^{(V,k)} \leftarrow$ RelationalFusion$^k(G^P, \mathbf{H}^{(V,k-1)}, \mathbf{H}^{(E,k-1)})$
        $\mathbf{H}^{(B',k-1)} \leftarrow$ Join$(\mathbf{H}^{(V,k-1)}, \mathbf{H}^{(B,k-1)})$
        $\mathbf{H}^{(E,k)} \leftarrow$ RelationalFusion$^k(G^D, \mathbf{H}^{(E,k-1)}, \mathbf{H}^{(B',k-1)})$
        $\mathbf{H}^{(B,k)} \leftarrow$ FeedForward$^k(\mathbf{H}^{(B,k-1)})$
    return $\mathbf{H}^{(V,K)}$, $\mathbf{H}^{(E,K)}$, $\mathbf{H}^{(B,K)}$
function Join($\mathbf{H}^V$, $\mathbf{H}^E$)
    for all $b \in B$ do
        let $b = \big((u,v),(v,w)\big)$
        $\mathbf{H}^{B'}_b \leftarrow \mathbf{H}^B_b \oplus \mathbf{H}^V_v$
    return $\mathbf{H}^{B'}$
```

With the relational fusion operator and its components defined in Sections 4.3 to 4.5, we proceed to explain how forward propagation is performed through the relational fusion layers (introduced in Section 4.2) of an RFN. The forward propagation algorithm is shown in Algorithm 2. Starting from the input encoding of node, edge, and between-edge attributes, each layer k transforms the node, edge, and between-edge representations emitted from the previous layer. The node representations are transformed using node-relational fusion. This is done by invoking the RelationalFusion function at line 4 with the primal graph, and the node and edge representations from the previous layer as input.

Edge-relational fusion is performed at lines 5-6 to transform the edge representations from the previous layer. In line 5, the node and between-edge representations from the previous layer are joined using the JOIN function (defined at lines 11-16) to capture the information from both sources that describe the relationships between two edges. On line 6, RELATIONALFUSION is invoked again, now with the dual graph (indicating relationships between edges), and the edge representations from the previous layer. The last input is the joined node and between-edge representations also from the previous layer. The between-edge representations are transformed using a single feed-forward operator at line 7.

The number of layers in an RFN determines which nodes, edges, and between-edges influence the output representations. Each layer aggregates information from the relations to first-order node and edge neighbors during edge- and node-relational fusion, respectively. Thus, a K-layer RFN aggregates information up to a distance K from nodes in the primal graph and edges in the dual graph, respectively.

Once the input representations have been propagated through all the layers, the final node, edge, and between-edge representations are output on line 9. These three outputs allows the relational fusion network to be jointly optimized for node, edge, and between-edge predictions, e.g., when the network is operating in a multi-task learning setting. However, if only one or two outputs are desired, the superfluous operations in the last layer can be skipped to save computational resources. For instance, propagation of node and between-edges can be skipped by replacing line 4 and line 7 with $\mathbf{H}^{(V,k)} = \mathbf{H}^{(V,k-1)}$ and $\mathbf{H}^{(B,k)} = \mathbf{H}^{(B,k-1)}$, respectively, when $k = K$.

4 Experimental Evaluation

We evaluate our method on two machine learning tasks: driving speed estimation and speed limit classification. These tasks represent a regression task and a classification task, respectively. We evaluate our method on both within-network prediction and cross-network prediction.

In the within-network setting, models are trained and evaluated on the road network of the same municipality. In the cross-network setting, models are trained on the road network of one municipality, but tested on another, unseen road network. For all tasks and settings, we compare the results of our method against a number of baselines, including state-of-the-art GCNs. Finally, we conduct a case study to investigate the behavior of RFNs under conditions of volatile homophily.

Table 1: Road network sizes.

	AAL	*BRS*	*CPH*
No. of Nodes ($\lvert V\rvert$)	16 294	5 889	10 738
No. of Edges ($\lvert E\rvert$)	35 947	13 073	26 117
No. of Between-Edges ($\lvert B\rvert$)	94 718	34 428	72 147

4.1 Data Set

We extract the spatial representation of the Danish municipalities of Aalborg (AAL), Brønderslev (BRS), and Copenhagen (CPH) from OSM [14] and convert them to their primal and dual graph representations as described in Section 2. The sizes of these networks can be seen in Table 1. We combine the OSM data with a zone map from the Danish Business Authority[1]. From these two data sources, we derive 3 node features, 16 edge features, and 5 between-edge features. See Appendix A for further details on the feature derivation process.

For the driving speed estimation task, we use a dataset of observed driving speeds, each matched to a road segment, derived from a set of vehicle trajectories collected between January 1 2012 and December 31 2014 [15]. Each such observed driving speed corresponds to a traversal in this set of trajectories [15]. Further details on the vehicle trajectory data can be found elsewhere [15].

For speed limit classification, we use 19 510 speed limits collected from the OSM data and additional speed limits collected from the municipality of Aalborg. This dataset is highly imbalanced, e.g., some speed limits are several thousands of times more frequent than others.

We split speed limits and driving speeds into training, validation, and test sets. For the driving speeds, the split is temporal s.t. the oldest observations are in the training set, the newest are in the test set, and the driving speeds in the validation set cover a period in-between. Table 2 shows the total number of driving speeds and speed limits with the proportion between training, validation, and testing sets.

4.2 Algorithms

We use the following GCN baselines for comparison in the experiments: (1) the Max-Pooling variant of GraphSAGE [11] and the (2) Graph Attention Network [12]. In addition, we include two non-GCN baselines: (3) a regular Multi-Layer Perceptron (MLP), and (4) a Grouping Estimator. The Grouping Estimator is a simple baseline that groups all road segments depending on

[1] `https://danishbusinessauthority.dk/plansystemdk`

Table 2: Dataset sizes and splits.

	Size	*Train / Val / Test*
Driving Speeds (AAL)	8 675 599	0.46 / 0.21 / 0.32
Driving Speeds (BRS)	847 963	0.00 / 0.00 / 1.00
Speed Limits (AAL)	19 510	0.50 / 0.23 / 0.27
Speed Limits (CPH)	17 824	0.00 / 0.00 / 1.00

their road category and on whether they are within a city zone or not. At prediction time, the algorithm outputs the mean (for regression) or mode (for classification) of the group a particular road segment belongs to.

In our experiments, we include four variants of the RFN that combines different relational aggregators and fusion functions. We examine attentional (A) aggregation and non-attentional (N) aggregation together with additive (A) and interactional fusion (I). The non-attentional computes an unweighted mean over the relational representations, i.e., all neighbors have the same attention weight. Each variant is denoted by combination their aggregator and fusion function acronyms, e.g., RFN-A+I.

4.3 Experimental Setup

The GraphSAGE and GAT models are run on the dual graph representations of the road network s.t. they learn edge representations directly. All models are two-layer models and use the ELU [16] activation function on the first layer. ReLU [17] and softmax are used on the last layer for driving speed estimation and speed limit classification, respectively. The ReLU [17] activation function is used in the GraphSAGE pooling operation for both tasks. We select layer sizes, learning rates, and the number of GAT attention heads by evaluating different hyperparameter configurations on the validation sets in a grid search and selecting the best-performing configuration. Each configuration is repeated ten times, and the configuration with the highest mean performance is selected for final evaluation on the test set. Appendix B provide additional details on the hyperparameter selection process and documents the hyperparameters are used for each model in our experiments.

All neural network algorithms are implemented using the MXNet[2] deep learning library. We make our implementation of the relational fusion networks publicly available[3].

[2] `https://mxnet.incubator.apache.org/`

[3] https://github.com/TobiasSkovgaardJepsen/relational-fusion-networks

Model Training and Evaluation We initialize the weights of all neural network models using Xavier initialization [18] and train the models using the ADAM optimizer [19] in (mini-)batches of 256 segments. For training efficiency, the batches are pre-computed in a stratified manner s.t. the distributions of road categories and speed limits—for speed limit classification and driving speed estimation, respectively—approximate the distributions of the full training set. The mini-batches are shuffled after each epoch.

We train all models for 20 and 30 epochs for driving speed estimation and speed limit classification, respectively, on Aalborg. For the cross-network setting, we use Brønderslev and Copenhagen for driving speed estimation and speed limit classification, respectively. For speed limit classification, the training set is randomly oversampled to address the class imbalance and regularize the models with early stopping.

For speed limit classification, we train the models using the binary cross entropy loss and evaluate the models using the F_1 macro score. For driving speed estimation, we train the models using a per-segment mean squared loss $\sum_{y \in Y} \frac{(\hat{y}-y)^2}{|Y|}$, where Y is the set of observed driving speeds associated with the segment, and $\hat{y}$ is the model's estimate of the driving speed of segment. The per-batch loss is the mean per-segment loss.

Driving speed estimation models are evaluated using a per-segment Mean Absolute Error (MAE): $|\hat{y} - \bar{Y}|$, where $\bar{Y} = \sum_{y \in Y} \frac{y}{|Y|}$ is the mean recorded speed for the segment. The final error of a model is the mean over all the per-segment errors. However, the value of $\bar{Y}$ is highly sensitive to outliers. We therefore exclude segments with fewer than ten observations when calculating the final error.

Using the per-segment losses and errors to train and evaluate models, respectively, on the driving speed estimation task avoids a bias towards models that perform well on popular road segments in the data set.

4.4 Results

We train ten models for each algorithm and task on Aalborg Municipality experiments for each model. We use these models for both within-network and cross-network inference and report the mean performance with standard deviations in Table 3. When reading Table 3, note that low values and high values are desirable for driving speed estimation and speed limit classification, respectively.

The GAT algorithm can become unstable during training [12] and we observed this phenomenon on one out of the ten runs on the driving speed estimation task. The algorithm did not converge on this run and is therefore excluded from the results shown in Table 3. In addition, none of the models give reasonable results on cross-network speed limit classification where top-

Table 3: Algorithm performance on Driving Speed Estimation (DSE) and Speed Limit Classification (SLC) on AAL and BRS.

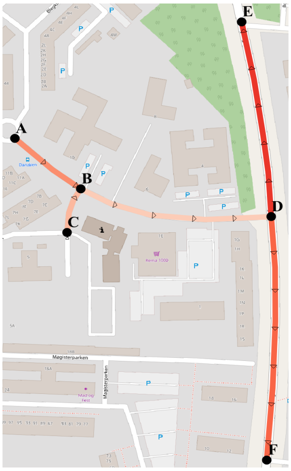
	DSE		SLC
Algorithm	*AAL*	*BRS*	*AAL*
GP	11.026	12.715	0.356
MLP	10.160 ± 0.119	12.659 ± 0.265	0.443 ± 0.027
GAT	9.548 ± 0.151	12.119 ± 0.342	0.442 ± 0.018
GraphSAGE	8.960 ± 0.115	11.292 ± 0.293	0.432 ± 0.014
RFN-N+A	7.685 ± 0.189	9.382 ± 0.447	0.500 ± 0.011
RFN-A+A	7.440 ± 0.133	9.078 ± 0.080	0.518 ± 0.022
RFN-N+I	6.911 ± 0.080	8.823 ± 0.196	0.507 ± 0.012
RFN-A+I	**6.797 ± 0.124**	**8.587 ± 0.238**	**0.535 ± 0.014**

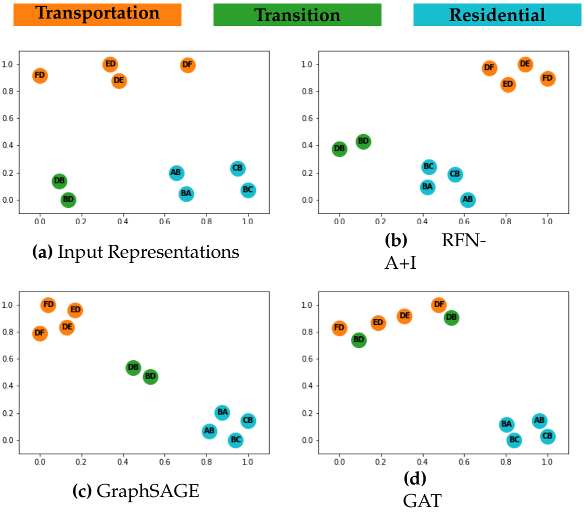
performance across all models is reduced to almost a third. Hence, we also exclude these results from Table 3.

Within-Network Inference

As shown in Table 3, the RFN variants outperform all baselines on both within-network driving speed estimation and within-network speed limit classification. The best RFN variant outperforms the state-of-the-art graph convolutional approaches, i.e., GraphSAGE and GAT, by 32% and 40%, respectively, on the driving speed estimation task. On the speed limit classification task, the best RFN variant outperforms GraphSAGE and GAT by 24% and 21%, respectively.

Table 3 shows that the RFN variants achieve similar performance, but the attentional variants are superior to their non-attentional counterparts that use the same fusion function. In addition, the interactional fusion function appears to be strictly better than the additive fusion function. Interestingly, GraphSAGE and GAT fail to outperform MLP on the speed limit classification task. The MLP classifies road segments independent of any adjacent road segments. Unlike the RFN variants, it appears that GraphSAGE and GAT are unable to effectively leverage the information from adjacent road segments. This supports our discussion in Section 1 of the problems with direct inheritance during neighborhood aggregation in the context of road networks.





(a) Input Representations

(b) RFN-A+I

(c) GraphSAGE

(d) GAT

Fig. 5: Ground truth driving speeds of the Danalien region in Aalborg, Denmark. Triangles indicate direction, black dots mark nodes, and color indicates speed: the darker, the faster.

Fig. 6: t-SNE visualizations of the edge representations in the Danalien case (a) before and (b, c, d) after the first graph convolution in different models.

Cross-Network Inference

Table 3 shows that the ranking of the models remain the same on driving speed estimation in the cross-network setting: all RFN variants outperform the baselines and the RFN-A+I remains the best-performing variant. All models suffer a performance loss, but the performance loss of the RFN variants is smaller than that of the neural network baselines. The best-performing baseline, GraphSAGE, has an error increase of 2.332 km/h. In comparison, the best-performing RFN variant, RFN-A+I, has a 23% smaller performance loss with an error increase of 1.790 km/h.

The small performance loss of the RFN variants compared to the neural network baselines on cross-network driving speed estimation suggests that (1) RFNs may learn transferable knowledge that generalizes well to other road networks and (2) that they are robust to substantial changes to the road network structure. However, we did not observe these tendencies on cross-network speed limit classification where all models achieved poor performance. This suggests that knowledge transferability is highly task-dependent and that some tasks may require sophisticated transfer learning methods.

4.5 Case Study: Danalien

Table 3 shows that RFNs generally perform better than the GraphSAGE and GAT baselines. However, it is unclear whether RFNs are capable of separating areas that are dissimilar, but internally homophilic, as intended. We have therefore examined the RFN's separation capabilities in areas exhibiting volatile homophily. For comparison, we have also examined the separation capabilities of the GraphSAGE and GAT algorithms. For the sake of brevity, we present only our case study on the Danalien intersection that was introduced in Figure 1. In this study, we consider a larger part of the residential area, shown in Figure 5. The Danalien case represents a quite typical scenario: a residential area that is connected to the rest of the city through a larger road. We have also examined a more extreme, but less typical case—cf. Appendix C.

We select the RFN, GraphSAGE, and GAT models with the best validation performance. For these models, we compare the representations of the road segments in the Danalien case before and after applying the first graph convolutional layer by projecting these representations into two dimensions using t-SNE [20] and normalizing the dimensions to be between 0 and 1. These representations are visualized in Figure 6.

The segments AB, BA, BC, and CB in Figure 5 form a residential area, and segments DE, ED, DF, and FD form a transportation area. In addition, segments BD and DB are transition segments that must be traversed to transition between the two areas. The residential, transportation, and transition segments are colored differently in the visualizations shown in Figure 6.

Interestingly, the graph convolution in the different models have different effects on the relative position of the segments in the representation space. In the input representation, illustrated in Figure 6a, the road segments are grouped with road segments of the same category. As shown in Figure 6b, the first graph convolution of the RFN-A+I model, makes these groups tighter and separates the residential and transportation segments further. Although not included in Figure 6, the other RFN variants have a similar effect on the position of the road segments in the representation space.

As shown in Figure 6c, the GraphSAGE model tightens the groups from Figure 6a like the RFN-A+I model, but places the transition segments roughly equi-distant from the residential and transportation clusters. The transition segments are more similar in terms of driving speed to the residential segments, than the transportation segments Therefore, this behavior of the GraphSAGE algorithm is expected since the GraphSAGE model inherits neighbor representations directly and indiscriminately (i.e., all neighbors contribute equally to the aggregate without an attention mechanism). Since the transition segments have the same number of transportation and residential segments as neighbor segments, the representations of the transition segments is between the two other categories in the representation space.

The GAT algorithm functions in a very similar manner to the GraphSAGE algorithm, except that it can select which neighbor segments to inherit neighbor representations from using its attention mechanism. The logic of this attention mechanism is shared across neighbors that has similar features (or belong to the same category in the Danalien example). When computing a new representation for the transition segments, the GAT model will therefore give preference to the transportation segments, give preference to the residential segments, or give no preference at all. As illustrated in Figure 6d, the GAT model gives preference to the transportation segments both when computing representations for the transition segments and when computing representations for the transportation segments. Thus, the GAT model groups the transition segments with the transportation segments, despite the transition segments being more similar to the residential segments in terms of driving speed.

As indicated by the tightness of the transition and transportation cluster in Figure 6d, the preference for the transportation segments is substantial when computing transition segment representations. We suspect that this behavior is attributed to two causes. First, the relative rarity of transition segments causes poor predictive performance on such segments to have little impact on the total loss. Second, road segments typically have at least one neighbor segment with a similar feature representation (e.g., belongs to the same category) and therefore such segments generally contribute little new information to distinguish a road segment of, e.g., the transition category from another road segment of the transition category.

In conclusion, we observe that the RFN model is able to create separate the different segment categories in the Danalien case better than the GAT and GraphSAGE models are, which suggests that the RFN architecture is more robust to volatile homophily.

5 Related Work

GCNs can broadly be categorized as either spatial or spectral.

Spectral GCNs [8–10, 13] are based on graph signal processing theory, which makes them theoretically well-founded although only for undirected graphs. This makes them ill-suited for road networks that often contain one-way streets and where, e.g., the driving speed on a road segment may heavily depend upon the direction of travel. In addition, spectral GCNs are transductive and rely on the specific graph structure on which it was trained and therefore does not support insertion and removal of edges to the network. This is problematic in road networks, where construction work frequently changes the network structure when, e.g., a motorway is built or a residential area is expanded.

Unlike spectral GCNs, our proposed method explicitly considers road segment driving directions and is robust to substantial changes in the road network structure. As we demonstrate in our experiments, our proposed method may make predictions on a different road network than it was trained on with only a minor decrease in predictive accuracy.

Our proposed method belongs to the class of spatial GCNs [11, 12], which are inductive methods. Like our proposed method, spatial GCNs support directed graphs and changes in the network structure. However, neither state-of-the-art spatial [11, 12] nor spectral GCNs [8–10, 13] support only node attributes. In addition, both classes of GCNs implicitly assume gradual changes in, e.g., driving speeds, when making predictions over the network and are thus ill-equipped to address the volatile homophily present in road networks. In contrast, our method can incorporate edge attributes and between-edge attributes, in addition to node attributes, and has built-in mechanisms to address volatile homophily in the input road network.

Research in GCNs architectures for road networks has thus far focused on extending GCNs architectures to solve specific problems with temporal dependencies [3, 21, 22]. In contrast, our work explores the spatial and structural aspects of GCN architectures and we present a novel GCN architecture that is designed to be generally applicable for machine learning on road networks. In addition, previous work has focused on spectral GCN architectures, whereas our method is a spatial GCN architecture. As discussed above, spatial GCNs (such as our proposed method) have several advantages over spectral GCNs, and we expect that our method can be used as a drop-in replacement for the spectral graph convolutions used in previous work.

Previous work has studied homophily in road networks [5]. Like us, they find that homophily is expressed differently in road networks—which we refer to as volatile homophily—but study this phenomenon in the context of transductive network embedding methods, whereas we focus on inductive GCNs. Unlike us, they do not propose a method to address the challenges of volatile homophily in road networks.

A preliminary four-page conference version of this paper presents the RFN-N+A variant [23]. The present version proposes an additional fusion function, InteractionalFuse, and an attentional aggregation function for RFNs. These extensions yield substantial improvements in our experiments. In addition, this version includes an evaluation of the RFN variants' robustness to changes in road network structure in cross-network inference, and it includes a detailed case study on the behavior of RFNs graph convolutions under conditions of volatile homophily.

6 Conclusion

We propose a method for machine learning on road networks. We argue that many built-in assumptions of existing proposals do not apply in this setting, in particular the assumption of smooth homophily. In addition, state-of-the-art GCNs can leverage only one source of attribute information, whereas we identify three sources of attribute information in road networks: intersection, segment, and between-segment attributes. We therefore propose the *Relational Fusion Network (RFN)*, a novel type of GCN for road networks.

We compare different RFN variants against a range of baselines, including two state-of-the-art GCN algorithms on both within-network and cross-network driving speed estimation and speed limit classification. We show that RFNs are able to outperform the GCN baselines by 32–40% and 21–24% on within-network driving speed estimation and speed limit classification, respectively. Our experiments suggest that the assumptions of GCNs do not apply to road networks: on within-network speed limit classification, the GCN baselines fail to outperform a multi-layer perceptron that does not incorporate information from adjacent edges. In addition, we show that RFNs generalize better on cross-network driving speed estimation and suffered a 23% smaller performance loss compared to the best performing neural network baseline. We also show that RFNs are more robust to volatile homophily than state-of-the-art GCNs in a case study.

We use relatively small road networks, and we therefore find it prudent to comment on the scalability of RFNs. Both RFNs and GCNs have a worst time complexity of $O(|E|d^K)$, where $d = \max(\{|N(e) \mid e \in E\})$ is the max node degree of the dual graph and K is the number of layers.

Several methods have been proposed to improve GCN forward propagation efficiency. First, by performing forward propagation only on the subnetwork necessary to compute an output for the road segments $E' \subseteq E$ for which an output is required [11]. Second, the global gradient can be approximated without recursive neighborhood expansion [24–26]. To the best of our knowledge, RFNs are compatible with these methods, and they can reduce the complexity to $O(|E'|dK)$ and $O(|E'|d^K)$ during training and inference, respectively.In practice, $|E'|$ is often substantially smaller than $|E|$. For instance, $|E'| = 5\,266$ and $|E| = 35\,947$ during testing for speed limit classification in Aalborg. Hence, we expect that RFNs can scale to country-sized road networks.

6.1 Future Work

Our experiments show that graph convolutional networks designed for road networks can yield substantial improvements in predictive performance. RFNs focus on the properties of volatile homophily, but do not explicitly utilize the spatial representation of road networks beyond road segment connectivity. We expect that a more complete utilization of the spatial representations of road networks may yield further improvements over GCNs for general networks.

The RFN variants we propose in this work cannot explicitly exploit temporal aspects of the data, but many tasks, e.g., driving speed estimation, are time-dependent. Extending RFNs to learn temporal road network dynamics is therefore an important future direction.

Current spatio-temporal GCN approaches focus on solving specific tasks by applying a temporal layer atop the spatial GCN layer [3, 21, 22]. As mentioned in Section 5, we expect that the RFN can be used as a drop-in replacement for the GCN in such methods, but the RFN architecture enables novel approaches for capturing temporal dynamics. For instance, the relationships between two adjacent road segments may change during the day. RFNs may be able capture such changes, e.g., through time-dependent fusion functions that accept temporal inputs, thus making the behavior of the graph convolution itself time-dependent.

All models achieved reasonable results on within-network speed limit classification, but failed to generalize to an unseen road network on cross-network speed limit classification. However, all models also struggled to achieve the within-network results: the validation score could change drastically from epoch to epoch, causing us to employ early stopping as a regularization method in our experiments. We therefore expect performance to be generally poor in areas that have not been observed during training. The speed limit data is crowd-sourced and thus labels tend to be concentrated in densely-populated areas leaving other areas scarcely labeled.

The driving speeds follow a similar pattern as the speed limits, where popular areas tend to have many driving speed observations, but we do not observe as severe a decline in predictive performance for cross-network driving speed estimation. We hypothesize this is because a larger part of the road network is seen during training on this task, and because the decision boundaries in classification tasks make classification models more sensitive to changes in the input. Investigating when knowledge learned from one part of a road network can be transferred to another part represents an interesting direction for future work.

Finally, we have only investigated the use of RFNs for machine learning on road networks. However, RFNs may be useful in other important application areas, e.g., multi-modal data fusion [27, 28] or machine learning on biological, citation, and social networks [11, 12]. Investigating the applicability of RFNs and their components to other types of networks and tasks present an interesting future direction.

References

[1] Y. Lv, Y. Duan, W. Kang, Z. Li, and F. Wang, "Traffic Flow Prediction With Big Data: A Deep Learning Approach," *IEEE Transactions on ITS*, vol. 16, no. 2, pp. 865–873, 2015.

[2] Z. Zheng, Y. Yang, J. Liu, H. Dai, and Y. Zhang, "Deep and Embedded Learning Approach for Traffic Flow Prediction in Urban Informatics," *IEEE Transactions on ITS*, vol. 20, no. 10, pp. 3927–3939, 2019.

[3] B. Yu, H. Yin, and Z. Zhu, "Spatio-Temporal Graph Convolutional Networks: A Deep Learning Framework for Traffic Forecasting," in *Proc. of IJCAI*, 2017, pp. 3634–3640.

[4] D. Zang, J. Ling, Z. Wei, K. Tang, and J. Cheng, "Long-Term Traffic Speed Prediction Based on Multiscale Spatio-Temporal Feature Learning Network," *IEEE Transactions on ITS*, vol. 20, no. 10, pp. 3700–3709, 2019.

[5] T. S. Jepsen, C. S. Jensen, T. D. Nielsen, and K. Torp, "On Network Embedding for Machine Learning on Road Networks: A Case Study on the Danish Road Network," in *Proc. of Big Data*, 2018, pp. 3422–3431.

[6] Y. Li, K. Fu, Z. Wang, C. Shahabi, J. Ye, and Y. Liu, "Multi-task Representation Learning for Travel Time Estimation," in *Proc. of KDD*. ACM, 2018, pp. 1695–1704.

[7] P. He, G. Jiang, S. Lam, and D. Tang, "Travel-Time Prediction of Bus Journey With Multiple Bus Trips," *IEEE Transactions on ITS*, vol. 20, no. 11, pp. 4192–4205, 2019.

[8] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral Networks and Locally Connected Networks on Graphs," 2014, p. 14 pp.

[9] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering," in *Proc. of NIPS*, 2016, pp. 3844–3852.

[10] T. N. Kipf and M. Welling, "Semi-Supervised Classification with Graph Convolutional Networks," in *Proc. of ICLR*, 2017, p. 14 pp.

[11] W. L. Hamilton, R. Ying, and J. Leskovec, "Inductive Representation Learning on Large Graphs," in *Proc. of NIPS*, 2017, pp. 1024–1034.

[12] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph Attention Networks," in *Proc. of ICLR*, 2018, p. 12 pp.

[13] R. Levie, F. Monti, X. Bresson, and M. M. Bronstein, "CayleyNets: Graph Convolutional Neural Networks With Complex Rational Spectral Filters," *IEEE Transactions on Signal Processing*, vol. 67, no. 1, pp. 97–109, 2019.

[14] OpenStreetMap contributors, "Planet dump retrieved from https://planet.osm.org," 2014.

[15] O. Andersen, B. B. Krogh, and K. Torp, "An Open-source Based ITS Platform," in *Proc. of MDM*, vol. 2, 2013, pp. 27–32.

[16] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs) ," in *Proc. of ICLR*, 2016, p. 14 pp.

[17] X. Glorot, A. Bordes, and Y. Bengio, "Deep Sparse Rectifier Neural Networks," in *Proc. of AISTATS*, 2011, pp. 315–323.

[18] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. of AISTATS*, 2010, pp. 249–256.

[19] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. of ICLR*, 2015, p. 15 pp.

[20] L. van der Maaten, "Accelerating t-sne using tree-based algorithms," vol. 15, pp. 3221–3245, 2014.

[21] J. Hu, C. Guo, B. Yang, and C. S. Jensen, "Stochastic Weight Completion for Road Networks using Graph Convolutional Networks," in *Proc. of ICDE*, 2019, pp. 1274–1285.

[22] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-GCN: A Temporal Graph Convolutional Network for Traffic Prediction," *IEEE Transactions on ITS*, 2019.

[23] T. S. Jepsen, C. S. Jensen, and T. D. Nielsen, "Graph Convolutional Networks for Road Networks," in *Proc. of SIGSPATIAL*, 2019, pp. 460–463.

[24] J. Chen, T. Ma, and C. Xiao, "FastGCN: Fast Learning with Graph Convolutional Networks via Importance Sampling," in *Proc. of ICLR*, 2018, p. 15 pp.

[25] W. Huang, T. Zhang, Y. Rong, and J. Huang, "Adaptive sampling towards fast graph representation learning," in *Proc. of NIPS*, 2018, pp. 4558–4567.

[26] D. Zou, Z. Hu, Y. Wang, S. Jiang, Y. Sun, and Q. Gu, "Layer-Dependent Importance Sampling for Training Deep and Large Graph Convolutional Networks," in *Proc. of NIPS*, 2019, pp. 11 247–11 256.

[27] Z. Li, B. Guo, Y. Sun, Z. Wang, L. Wang, and Z. Yu, “An attention-based user profiling model by leveraging multi-modal social media contents,” in *Cyberspace Data and Intelligence, and Cyber-Living, Syndrome, and Health*, 2019, pp. 272–284.

[28] B. Guo, Y. Ouyang, C. Zhang, J. Zhang, Z. Yu, D. Wu, and Y. Wang, “CrowdStory: Fine-Grained Event Storyline Generation by Fusion of Multi-Modal Crowdsourced Data,” *IMWUT*, vol. 1, no. 3, pp. 1–19, 2017.

Appendices

A Feature Derivation

We describe here which node, edge, and between-edge features are used in our experiments and how they are derived from node, edge, and between-edge attributes.

A.1 Node Attributes

We derive node attributes using the zone map from the Danish Business Authority mentioned in Section 4.1. The map categorizes regions in Denmark as city zones, rural zones, and summer cottage zones. We use a node attribute for each of the three zone categories to indicate whether or not an intersection belongs to the zone category or not. Each of these node attributes is encoded as a binary value, either 0 or 1, yielding three features per node.

A.2 Edge Attributes

The OSM data includes categories for all road segments. There are nine different road segment categories in the OSM extract that we use for our experiments. In addition, the length of each road segment can be derived from its spatial representation. We use both road segment category and length as edge attributes.

Road segment categories are one-hot encoded into nine-dimensional vectors. Road segment lengths are encoded as continuous values that are scaled to the $[0;1]$ range to ensure that all features are on the same scale. Scaling road segment lengths into the same range as the other features makes training with the gradient-based optimization algorithm used in our experiments more stable. The encoding of road segment categories and lengths yields a total of ten edge features.

Finally, the node features of the source and target nodes of the edge are concatenated with these edge features, yielding 16 edge features in total. The concatenation of source and target node features offers the neural network baselines (MLP, GraphSAGE, GAT) access to both node and edge features, rather than just edge features.

A.3 Between-Edge Attributes

The between-edge attributes we consider are the turn direction and the turn angle.

The turn direction can take on four values: straight-ahead, left, right, and U-turn. We one-hot encode the turn-direction attribute into a four-dimensional vector. The turn angle takes on values between 0 and 180 degrees. We encode the turn angle attribute as a continuous value. As with road segment lengths, turn angles are scaled into the $[0;1]$ range. This yields five between-edge features in total.

B Hyperparameter Selection

As mentioned in Section 4.3, hyperparameter values for the neural network baselines are selected using a grid search over a set of hyperparameter configurations. For each algorithm, we select the hyperparameter configuration that achieves the best mean performance across ten runs. We explore learning rates $\lambda \in \{0.1, 0.01, 0.001\}$ in the grid search based on preliminary experiments. For GraphSAGE, GAT, and all RFN variants, we explored $d \in \{32, 64, 128\}$ output dimensionalities of the first layer. The MLP uses considerably fewer parameters than the other algorithms, and we therefore also explore larger hidden layer sizes $d \in \{128, 256, 512\}$ for a fair comparison.

B.1 RFN Variants

For driving speed estimation, we use L_2 normalization on the first layer. For speed limit classification, we found that training became more stable when using L_2 normalization on both layers.

B.2 GraphSAGE

GraphSAGE uses a pooling network to perform neighborhood aggregation. For each layer in GraphSAGE, we set the output dimension of the pooling network to be double the output dimension of the layer in accordance with the authors' experiments [11].

By definition, GraphSAGE applies L_2 normalization on the output of each layer. However, we omit L_2 normalization on the last (second) layer of GraphSAGE models for driving speed estimation; otherwise, the output cannot exceed a value of one, resulting in poor performance.

B.3 GAT

The GAT algorithm uses multiple attention heads that each output d features at the first hidden layer. These features are subsequently concatenated, thus yielding an output dimensionality of $h \cdot d$, where h is the number of attention heads. Based on past work [12], we explore different values of $h \in \{1, 2, 4, 8\}$

during the grid search and use the same number of attention heads for both layers. The concatenation makes the GAT network very time-consuming to train when both h and d are large. We therefore budget these parameters s.t. $h \cdot d \leq 256$, corresponding to, e.g., a GAT network with 64 output units from 4 attention heads.

B.4 Selected Hyperparameters

The hyperparameters selected used for evaluation on the test set is shown in Table 4 for the Driving Speed Estimation (DSE) and Speed Limit Classification (SLC) tasks.

Table 4: Best model hyperparameters found in the grid search.

	Task					
Algorithm	DSE			SLC		
MLP	$d = 128$	$\lambda = 0.01$		$d = 128$	$\lambda = 0.1$	
GAT	$d = 32$	$\lambda = 0.01$	$h = 8$	$d = 32$	$\lambda = 0.001$	$h = 8$
GraphSAGE	$d = 64$	$\lambda = 0.01$		$d = 64$	$\lambda = 0.001$	
RFN-N+A	$d = 32$	$\lambda = 0.01$		$d = 64$	$\lambda = 0.1$	
RFN-A+A	$d = 32$	$\lambda = 0.01$		$d = 32$	$\lambda = 0.1$	
RFN-N+I	$d = 32$	$\lambda = 0.01$		$d = 32$	$\lambda = 0.01$	
RFN-A+I	$d = 32$	$\lambda = 0.01$		$d = 64$	$\lambda = 0.01$	

C Case Study: Dall

The Danalien case discussed in Section 4.5 is a common case of volatile homophily, where there is still homophily s.t. each road segment has a similar neighbor segment. However, as discussed in Section 1, extreme cases of volatile homophily exists where there is no homophily. Such a case is illustrated in Figure 7 that concerns a resting area next to a motorway near the village of Dall. In the figure, segment CB is part of the resting area. Segments AB and CD is a motorway approach and exit, respectively. Finally, segment BC is a transition segment that allows access to and from the resting area. As indicated in Figure 7, segment CB has a substantially different driving speed than the other segments.

We again select the RFN, GraphSAGE, and GAT models and visualize the representations of the road segments of the Dall case before and after the first graph convolution in Figure 8 following the same procedure as in the Danalien case (cf. Section 4.5).

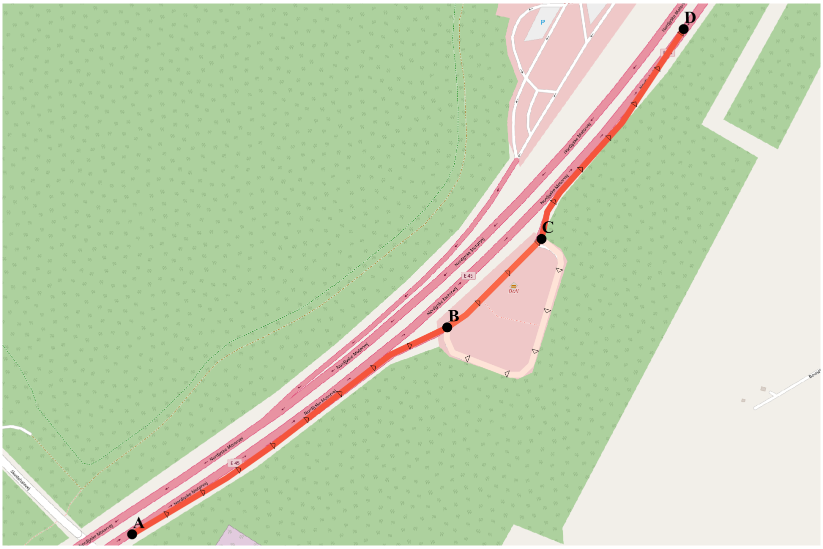
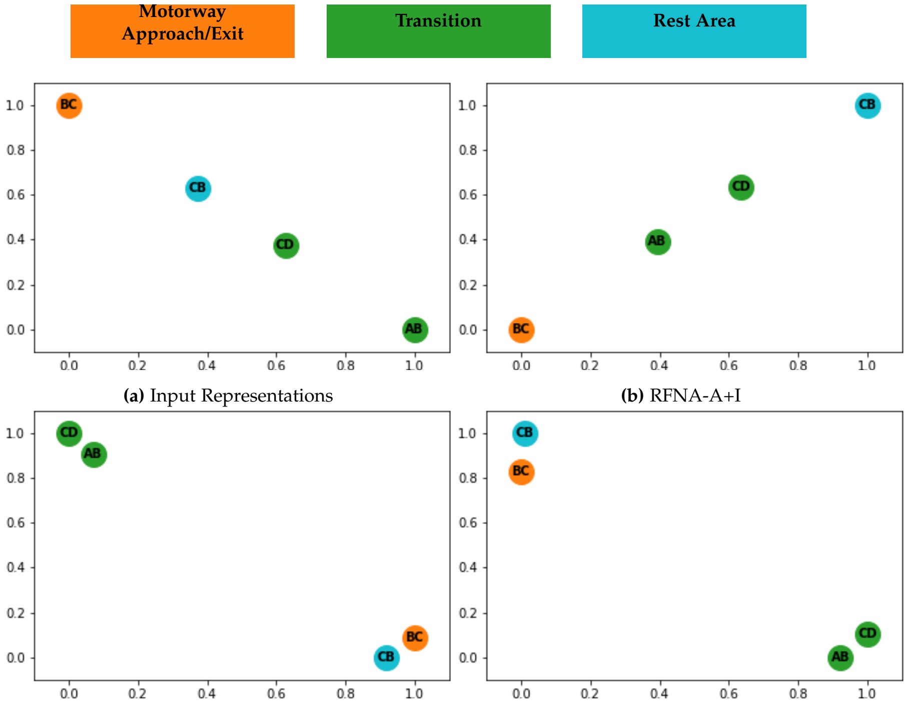


Fig. 7: Ground truth driving speeds of a resting area and its adjacent motorway segments near the village of Dall, Denmark. Triangles indicate direction, black dots mark nodes, and color indicates speed: the darker, the faster.



(a) Input Representations

(b) RFNA-A+I

(c) GraphSAGE

(d) GAT

Fig. 8: t-SNE visualizations of edge representations (a) before and (b, c, d) after the first graph convolution in different models.

Paper D

UniTE—The Best of Both Worlds: Unifying Function-Fitting and Aggregation-Based Approaches to Travel Time and Travel Speed Estimation

Tobias Skovgaard Jepsen
Aalborg University

Christian S. Jensen
Aalborg University

Thomas Dyhre Nielsen
Aalborg University

This paper is submitted to
ACM Transactions on Spatial Algorithms and Systems.

Abstract

Travel time or speed estimation are part of many intelligent transportation applications. Existing estimation approaches rely on either function fitting or aggregation and represent different trade-offs between generalizability and accuracy.

Function-fitting approaches learn functions that map feature vectors of, e.g., routes, to travel time or speed estimates, which enables generalization to unseen routes. However, mapping functions are imperfect and offer poor accuracy in practice. Aggregation-based approaches instead form estimates by aggregating historical data, e.g., traversal data for routes. This enables very high accuracy given sufficient data. However, they rely on simplistic heuristics when insufficient data is available, yielding poor generalizability.

We present UniTE that combines function-fitting and aggregation-based approaches into a unified framework that aims to achieve the generalizability of function-fitting approaches and the accuracy of aggregation-based approaches. An empirical study finds that an instance of UniTE can improve the accuracies of travel speed distribution and travel time estimation by 40–64% *and* 3–23%*, respectively, compared to using function fitting or aggregation alone.*

1 Introduction

Estimation of travel time or speed is central to many intelligent transportation applications [1] such as trajectory analysis [2], annotating road segments with travel times [3, 4], and traffic forecasting [5]. This often concerns routes in a road network, with road segments being special cases. For clarity of presentation, we thus assume that estimation is done for routes in the remainder of the paper, but our work is also relevant to other kinds of data, e.g., location-based travel speed forecasting using loop detectors [6]. Existing approaches to travel time and speed estimation can be categorised as either function-fitting [3–35] or aggregation-based [36–39] approaches.

Aggregation-based approaches use historical travel time or speed data to compute corresponding at estimates of the travel speed or time for routes, often for different time-of-day intervals. Estimates are typically given as histograms [36], e.g., as parameters that denote heights of bins in equi-width histograms. Function-fitting approaches fit a function f with parameters ψ that maps feature vector representations of routes, to travel time and speed estimates. The parameters ψ are found by using input-output pairs from historical data and minimizing the discrepancy between the mapping's output and the expected output. Function-fitting and aggregation-based approaches represent different trade-offs between estimation generalizability and accuracy.

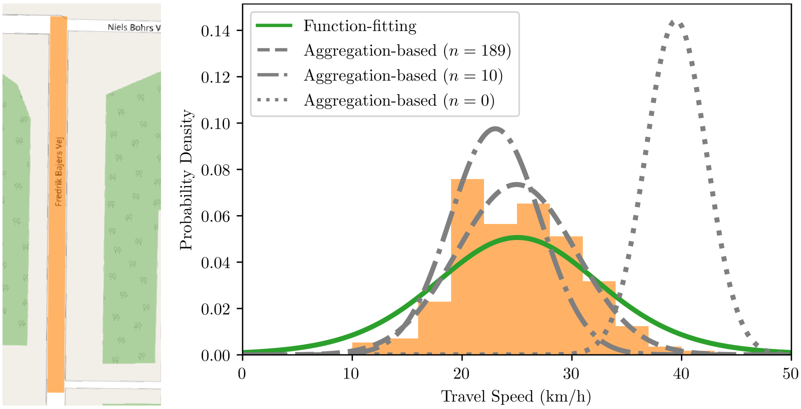


Fig. 1: A road segment (left) and its ground-truth and estimated travel-speed distributions during a time-of-day interval (right).

Aggregation-based approaches rely only on the sufficient availability of relevant and representative data when estimating the travel speed or time of, e.g., a route. Unlike function-fitting approaches, aggregation-based approaches use data directly at estimation time without an intermediary mapping function. If sufficient data is available, they can therefore achieve highly accurate point or distribution estimates with substantial less effort of implementation than function-fitting approaches. However, in country-sized road networks, such data is typically not available for all routes at all times of day [3, 40, 41], in which case aggregation-based approaches default to simple but inaccurate heuristics, resulting in poor generalizability.

The mapping function in function-fitting approaches can generalize to unseen routes, since only a feature vector representation of the route needs to be available. In principle, it is possible to fit a fine-granularity function to approach the accuracy of aggregation-based methods. However, this is not feasible in practice due to the substantial efforts required to do so which involves choosing a good structure for the mapping function [4, 6, 22, 31, 35], a good feature vector representation of the inputs [5, 31, 34, 35, 42], a strategy for handling imbalances of, e.g., road segment popularity, in the data [34, 35], and designing an appropriate loss function [4, 6, 31, 35]. As a result, the mapping function is imperfect: it may perform well in general, but poorly for particular routes.

The generalizability-accuracy trade-off between function-fitting and aggregation-based approaches is illustrated in Figure 1 using experimental results from our empirical study described in Section 5. The aggregation-based approach achieves the best fit when observing $n = 189$ samples, but when $n = 10$ underestimates the mean and variance since the data sample is not representative. When $n = 0$, the aggregation-based approach defaults to its heuristic which in this case overestimates the mean considerable and underestimates the variance even more. The function-fitting approach makes no use of data at estimation time and is unaffected by the representativeness of any available data. Instead, it extrapolates the travel speed distribution from other road segments with similar feature representations. In this case, it achieves a good estimation of the mean, but overestimates the variance.

Both function-fitting and aggregation-based approaches ignore the reality that, in practice, some areas of a road network are associated with plenty of data at all times, e.g., city centers, of day while other areas are associated with little-to-no data regardless of the time of day, e.g., some rural areas. Typically, the amount of available data from an area is proportional to population density. Handling this situation is necessary for fine-granularity travel time and speed estimation and requires flexibility that existing approaches lack: in areas with plenty of data an aggregation-based approach is ideal, but in areas with very little data a function-fitting approach may be necessary to achieve reasonable accuracy. To this end, we propose a Unifying approach to Travel time and speed Estimation (UniTE).

UniTE is a travel time and speed estimation framework that integrates (and complements) function-fitting and aggregation-based approaches. Using Bayesian probability theory, UniTE allows for gradual transition between function-fitting and aggregation based on the available data. In addition, we present an instance of the UniTE framework, Gaussian UniTE (G-UniTE), that models travel time or speed as a Gaussian variable with unknown mean and variance. The use of conjugate priors in G-UniTE allows for efficient computation of the posterior, easy implementation, and makes G-UniTE applicable to neural network learning using standard deep learning frameworks. Finally, we investigate the capabilities of the UniTE framework in an empirical study using G-UniTE. The study shows that UniTE can achieve 40–64% and 3–23% better performance in terms of travel speed distribution modeling and travel time point estimation, respectively, compared to using a function-fitting or aggregation-based approach alone. Furthermore, UniTE can achieve better generalizability than function-fitting approaches while maintaining similar or better accuracy to aggregation-based approaches regardless of data availability.

The remainder of the paper is structured as follows. Section 2 provides the necessary background on function-fitting and aggregation-based approaches, as well as graph modeling of road networks. Section 3 presents the UniTE framework and describe how UniTE can unify existing function-fitting and aggregation-based approaches. Section 4 presents the Gaussian instance of the UniTE framework, G-UniTE. Section 5 reports on the empirical study. Section 6 reviews related work, and Section 7 concludes and offers directions for future research.

2 Preliminaries

We now provide the necessary background on data modeling and existing approaches to travel time and speed estimation.

2.1 Data Modeling

For clarity, we present UniTE as well as existing function-fitting and aggregation-based approaches for routes, i.e., where travel times and travel speeds are estimated for routes in in a road network with road segments as a special case. UniTE is applicable to other kinds of data as well, to be discussed in Section 6.

Road Network Modeling

A road network is modeled as a directed graph $G = (V, E)$ where a vertex $v \in V$ represents an intersection or the end of a road, and an edge $e \in E$ represents a road segment. A route is a connected path $p = (e_1, \dots, e_n)$

where $e_i \in E$ for $1 \leq i \leq n$. In addition, a route can be mapped to a d-dimensional feature vector describing its characteristics using the mapping function ϕ. For brevity, we use the notation $\mathbf{p}$ to refer to the feature vector representation $\phi(p)$ of a route p. If p consists of one edge, i.e., $p = e$, we use the notation $\mathbf{e}$ instead.

Trajectory Modeling

Vehicle trajectories are sequences of time-stamped GPS locations, but can be map-matched to a road network modeled as a directed graph (as described in Section 2.1). Each map-matched trajectory is a sequence $TR = (tr_1, \ldots, tr_n)$ where $tr_i = (e_i, \tau_i, t_i)$ is a triple consisting of a road segment $e_i \in E$, an arrival time τ_i corresponding to the timestamp of the first recorded GPS location on road segment e_i, and t_i the travel time or travel speed recorded during the traversal of segment e_i. In some cases, the traversal of a road segment in a trip is inferred by the map-matching algorithm due to a lack of GPS data. In such cases, $t_i = \varnothing$ and $\tau_i = \varnothing$.

2.2 Existing Approaches

We now describe function-fitting and aggregation-based approaches to travel time or speed estimation for routes.

Function-Fitting Approaches

Function-fitting approaches [3–6, 8–31, 33, 34] assume that the relationship between the unknown future travel time or travel speed $\hat{t}_i$ when traversing route p_i at time τ_i can be modeled by a function f s.t. $\Pr(\hat{t}_i \mid p_i, \tau_i; \psi) = f(\mathbf{p}_i, \boldsymbol{\tau}_i; \psi)$ where ψ is the function parameters of f, $\mathbf{p}_i$ is the feature vector representation $\mathbf{p}_i$ of route p_i, and $\boldsymbol{\tau}_i$ is the vector representation of time τ_i.

As an example of a function f, if the probability density $\Pr(\hat{t}_i \mid p_i, \tau_i; \psi)$ is a uni-variate Gaussian, then a possible choice of f is $f(\mathbf{p}_i, \boldsymbol{\tau}_i; \psi) = \mathcal{N}(\hat{t}_i \mid \boldsymbol{\psi}_\mu(\mathbf{p}_i \oplus \boldsymbol{\tau}_i), \boldsymbol{\psi}_{\sigma^2}(\mathbf{p}_i \oplus \boldsymbol{\tau}_i))$ where $\oplus$ denotes vector concatenation and the parameters $\psi = \{\boldsymbol{\psi}_\mu, \boldsymbol{\psi}_{\sigma^2}\}$ consists of two real vectors. In this case, the parameters $\boldsymbol{\psi}_\mu$ and $\boldsymbol{\psi}_{\sigma^2}$ that are used to compute the mean and variance of the Gaussian, respectively.

To fit a function f to a set of n training trajectories, function-fitting approaches learn model parameters θ that maximize the conditional likelihood $\prod_{i=1}^{n} \Pr(\hat{t}_i \mid p_i, \tau_i; \psi) = f(\mathbf{p}_i, \boldsymbol{\tau}_i; \psi)$. By sharing model parameters ψ across training trajectories during optimization, function-fitting approaches can generalize to unseen routes. However, in practice, generalization is imperfect for non-trivial travel time and speed estimation tasks.

Aggregation-Based Approaches

Aggregation-based approaches [36–39] aim to learn model parameters θ_i for each route at time τ_i using a set of training trajectories. In other words, given a set of n training trajectories, these approaches solve n optimization problems.

Unlike function-fitting approaches, aggregation-based approaches do not optimize the posterior predictive directly. Instead, the model parameters θ_i are chosen s.t. they are the maximum a posteriori probability (MAP) estimate of $\Pr(\theta_i \mid \tau_i, \tilde{T}_i)$, where $\tilde{T}_i$ is a set of historical travel speed or time records that are typically collected from historical trajectories traversing route p_i during some interval based on time τ_i, e.g., the same time-of-week interval as τ_i.

By using distinct model parameters for each route set of training trajectories, aggregation-based approaches can learn a more accurate representation of the travel time or travel speed distribution of $\hat{t}_i$ provided sufficient historical records are available. However, in practice, it is unlikely that there is sufficient data for all routes at all times of day [3, 40, 41] in country-sized road networks. In such cases, aggregation-based approaches rely on simplistic heuristics to provide reasonable estimates for unseen routes, and they are prone to overfitting when only a few historical records are available.

3 A Unified Approach

We now present the proposed UniTE framework.

3.1 Framework

The primary goal of UniTE is to unify a function-fitting component with an aggregation-based component s.t. we can seamlessly and smoothly switch between the two to leverage their respective strengths. The two components should be integrated s.t. UniTE relies on the function-fitting component when no historical data is available or if the data is not representative, e.g., due to low availability. Conversely, we want UniTE to rely on the aggregation-based component when historical data abounds. To achieve this, UniTE adopts a Bayesian foundation that provides a solid theoretical basis for unifying function fitting and aggregation.

A conceptual model of UniTE is illustrated in Figure 2. In brief, we assume that the travel time or speed distribution of a route p_i at time τ_i follows a distribution with uncertain hyperparameters θ_i. The prior distribution of θ_i is computed as $\Pr(\theta_i \mid f(\mathbf{p}_i, \boldsymbol{\tau}_i; \psi))$ using a *prior function* f with function parameters ψ that are shared across all routes. The prior function f represents the function-fitting component of UniTE, and allows UniTE to estimate

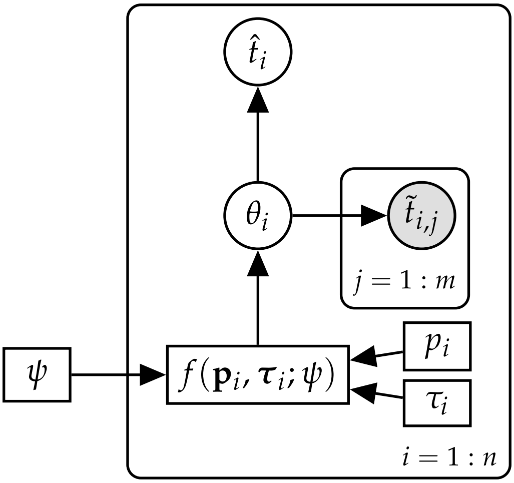


Fig. 2: The UniTE framework illustrated using plate notation.

prior distributions for routes even if no historical data is available at estimation time. This estimate is based on the travel time or travel speed distributions of similar routes at similar times. However, if historical records $\tilde{T}_i = \{\tilde{t}_{i,1}, \ldots, \tilde{t}_{i,m}\}$ are available, a posterior travel time or speed distribution $\Pr(t_i \mid \tilde{T}_i; \psi)$ for route p_i at time τ_i is computed. The computation of the posterior represents the aggregation-based component in UniTE.

3.2 The UniTE Objective

We now present the objective function used to train models within the UniTE framework.

Objective Function

Figure 2 depicts a generative model, i.e., a model that specifies how to generate the new records from the distribution $\Pr(\hat{t}_i \mid \theta_i)$ [43]. Generative models are usually trained by selecting parameters θ_i that maximize the joint likelihood [43]. However, training a generative model by maximizing the conditional likelihood is guaranteed to yield better estimations when the true travel time or travel speed distribution is different from the distribution family assumed by the model [44]. This is generally the case in practice, where, e.g., travel time distributions are highly complex [36]. In this work, we are interested in predictive performance and therefore maximize the conditional likelihood

$$\Pr(t_i \mid \tilde{T}_i, \theta_i) = \Pr(t_i \mid \tilde{T}_i, \mathbf{p}_i, \boldsymbol{\tau}_i; \psi) \tag{D.1}$$

across n training trajectories where t_i is the ground truth travel time or travel speed observed in the ith trajectory when traversing route p_i at time τ_i. The posterior predictive $\Pr(\hat{t}_i \mid \tilde{T}_i, \theta_i)$ equals the prior predictive $\Pr(\hat{t}_i \mid \theta_i)$ if no historical records are available as evidence, i.e., if $\tilde{T}_i = \emptyset$.

Regularizing Properties

The UniTE framework inherently addresses the issues of data imbalance issues of function-fitting approaches where they tend to fit best to frequently occurring types of, e.g., road segments. Because the UniTE framework is Bayesian, the UniTE objective in Equation D.1 is implicitly regularized s.t. the performance of the function-fitting component represented by prior function f is inversely proportional to the number of historical records available. In other words, the function-fitting component is trained to perform well in data-sparse situations.

The posterior predictive in Equation D.1, i.e.,

$$\Pr(\hat{t}_i \mid \tilde{T}_i, \theta_i) = \int_{\theta_i} \Pr(\hat{t}_i \mid \theta_i) \Pr(\theta_i \mid \tilde{T}_i) d\theta_i,$$

depends on the posterior distribution

$$\Pr(\theta_i \mid \tilde{T}_i) \propto \Pr(\theta_i) \prod_{j=1}^{m} \Pr(\tilde{t}_{i,j} \mid \theta_i). \tag{D.2}$$

As Equation D.2 shows, the number of factors in the product increases as m increases s.t. for large m, the factor $\Pr(\theta_i)$ has only a minor influence on the value of the final product. In other words, the importance of the prior distribution $\Pr(\theta_i)$ on the posterior distribution, and thus the posterior predictive, is inversely proportional to the number of historical records. As a consequence, the influence of the function-fitting component on the UniTE objective in Equation D.1 is largest when there are no historical records at all, and it gradually becomes less important if more historical records are available. Thus, by maximizing the conditional likelihood in Equation D.1, the function-fitting component is trained to perform well in data-sparse situations. This ensures that no explicit regularization of the function-fitting component, e.g., oversampling [34], is required to handle data imbalance issues.

3.3 Relation to Existing Approaches

UniTE can be viewed as a hybrid of function-fitting and aggregation-based approaches.

Hybrid Characteristics

The UniTE framework and it corresponding objective in Equation D.1 are hybrid in the sense that, like the function-fitting approaches, it fits a function f that maps input feature vector representations to a prior travel time or travel speed estimate by minimizing the discrepancy between the output of the mapping and the expected output. However, like the aggregation-based approaches, UniTE can also use historical records directly in the estimation process, i.e., without a typically imperfect intermediary mapping function, to adjust the prior estimate of its function-fitting component by computing the posterior. This adjustment allows UniTE, like aggregation-based approaches, to approximate a travel time or travel speed distribution at arbitrary precision given sufficient data and an appropriate choice of the distribution family for $\Pr(\hat{t}_i \mid \tilde{T}_i, \mathbf{p}_i; \theta)$.

A very appealing property of UniTE is that from a modeling capability perspective, UniTE models are capable of being at least as powerful as either their function-fitting or aggregation-based components. Specifically, a UniTE model can match the performance of its function-fitting component at arbitrary precision by expressing very high confidence in the prior. Similarly, a UniTE model can match the performance of its aggregation-based component at arbitrary precision by expressing very low confidence in the prior.

Integration with Existing Approaches

An important feature of the UniTE framework is that it is complimentary and integrable with existing approaches to travel time and speed estimation. Within the framework, existing function-fitting approaches provide the structure of the prior function f used to estimate the prior hyperparameters. We expect that most function-fitting approaches can be integrated with the UniTE framework with only minor modifications to the output layer and the objective function, depending on the choice of distribution for $\hat{t}$. Next, existing aggregation-based approaches are primarily concerned with the selection of historical records for aggregation. Aggregation-based approaches thus provide record selection strategies to construct the set of historical records $\tilde{T}_i$ in Equation D.1.

Remarks on Training

Unlike aggregation-based approaches, UniTE maximizes the conditional likelihood, and it is desirable that a function-fitting component (represented by prior function f) compensates for an aggregation-based component when insufficient historical data is available. In the extreme case, no data may be available. To simulate this situation during training, we recommend excluding the ground truth travel time or travel speed from the set of historical records, i.e., we recommend that $t_i \notin \tilde{T}_i$ in Equation D.1.

By excluding the ground truth travel time or travel speed from the set of historical records during the training, the function-fitting component must always contribute information missing from the set of historical records $\tilde{T}_i$ during training. Specifically, it must contribute information about the ground truth travel time or speed t_i to optimize the objective in Equation D.1.

4 Gaussian UniTE

UniTE is a framework for which many instantiations are possible. To study the prospects of UniTE analytically and empirically, we present one such instantiation, Gaussian UniTE (G-UniTE), that is both easy to implement and integrate with existing approaches.

As the name suggests, G-UniTE assumes that $\hat{t}_i \mid \theta_i$ follows a Gaussian distribution with parameters $\theta_i = (\mu_i, \lambda_i)$. The Gaussian assumption combined with the use of conjugate priors over the uncertain mean μ_i and precision λ_i allows the generally difficult-to-compute posterior predictive in Equation D.1 to be computed efficiently and in closed-form [45]. In addition, the closed-form computation of the posterior predictive is also differentiable. This enables the use of gradient-based optimization techniques that are commonly used in function-fitting approaches based on neural networks.

Note that UniTE is far more general than G-UniTE which is just one possible instantiation of the UniTE framework. Differentiability of the posterior predictive is a convenient property of G-UniTE but the UniTE framework is not restricted to gradient-based optimization.

4.1 Prior

Let $\Pr(\hat{t}_i \mid \mu_i, \lambda_i)$ denote the likelihood of a Gaussian distribution with mean μ_i and precision λ_i (or, equivalently, variance $\sigma_i^2 = \lambda^{-\frac{1}{2}}$). We adapt the work of Murphy [45] to our setting, and estimate μ_i and λ_i using the normal-gamma prior

$$\begin{aligned}\Pr(\mu_i, \lambda_i \mid \mathbf{p}_i; \psi) = NG(\mu_i, \lambda_i \mid \mu_{i,0}, \kappa_{i,0}, \alpha_{i,0}, \beta_{i,0}) = \\ \mathcal{N}(\mu_i \mid \mu_{i,0}, \frac{1}{\kappa_{i,0}\lambda_i}) Ga(\lambda_i \mid \alpha_{i,0}, \beta_{i,0}).\end{aligned} \quad \text{(D.3)}$$

Here, $NG(\mu_i, \lambda_i \mid \mu_{i,0}, \kappa_{i,0}, \alpha_{i,0}, \beta_{i,0})$ is the probability density function for a normal-gamma distribution with mean $\mu_{i,0}$, precision $\kappa_{i,0} > 0$, shape parameter $\alpha_{i,0} > 0$, and rate parameter $\beta_{i,0} > 0$. Similarly, $\mathcal{N}(\mu_i \mid \mu_{i,0}, \frac{1}{\kappa_{i,0}\lambda_i})$ and $Ga(\lambda_i \mid \alpha_{i,0}, \beta_{i,0})$ are probability density functions of a normal and a gamma distribution, respectively, over the mean μ_i and the precision λ_i. The hyperparameters of the prior, the *prior hyperparameters*, are output by the function f s.t. $f(\mathbf{p}_i;\psi) = \begin{bmatrix}\mu_{i,0} & \kappa_{i,0} & \alpha_{i,0} & \beta_{i,0}\end{bmatrix}$.

4.2 Posterior

After observing a sample of m historical records $\tilde{T}_i = \{\tilde{t}_{i,1}, \dots, \tilde{t}_{i,m}\}$, beliefs about μ and λ may change. Formally, the posterior distribution over μ_i and λ_i is given as [45]

$$\begin{aligned}\Pr(\mu_i, \lambda_i \mid \mathbf{p}_i, \tilde{T}_i;\theta) NG(\mu_i, \lambda_i \mid \mu_{i,m}, \kappa_{i,m}, \alpha_{i,m}, \beta_{i,m}) = \\ \mathcal{N}(\mu_i \mid \mu_{i,m}, \frac{1}{\kappa_{i,m}\lambda_{i,m}}) Ga(\lambda_i \mid \alpha_{i,m}, \beta_{i,m}),\end{aligned} \quad \text{(D.4)}$$

with *posterior hyperparameters*

$$\begin{aligned}\mu_{i,m} =& \frac{\kappa_{i,0}\mu_{i,0} + mM_{\tilde{T}_i}}{\kappa_{i,0} + m} \\ \kappa_{i,m} =& \kappa_{i,0} + m \\ \alpha_{i,m} =& \alpha_{i,0} + \frac{m}{2} \\ \beta_{i,m} =& \beta_{i,0} + \frac{1}{2} m S^2_{\tilde{T}_i} + \frac{1}{2} \frac{\kappa_{i,0} m (M_{\tilde{T}_i} - \mu_{i,0})^2}{\kappa_{i,0} + m},\end{aligned} \quad \text{(D.5)}$$

where $f(\mathbf{p}_i;\theta) = \begin{bmatrix}\mu_{i,0} & \kappa_{i,0} & \alpha_{i,0} & \beta_{i,0}\end{bmatrix}$, $M_{\tilde{T}_i} = \frac{\sum_{j=1}^{n} \tilde{T}_{i,j}}{m}$ is the sample mean, and $S^2_{\tilde{T}_i} = \frac{\sum_{j=1}^{m} (\tilde{t}_{i,j} - M_{\tilde{T}_i})^2}{m}$ is the biased sample variance. Note that if there is no data, i.e., $m = 0$, then the posterior hyperparameters are equal to the prior hyperparameters.

The regularizing properties of the UniTE framework discussed in Section 3.2 are reflected in the formulas for the posterior hyperparameters in Equation D.5. For instance, the posterior mean $\mu_{i,m}$ is a weighted mean of the prior mean $\mu_{i,0}$ and the mean of the historical records $M_{\tilde{T}_i}$ where $\mu_{i,0}$ has weight $\kappa_{i,0}$ and $M_{\tilde{T}_i}$ has weight m. Thus, the influence of the prior mean $\mu_{i,0}$ on the posterior mean $m_{i,m}$ diminishes as m increases. The remaining posterior hyperparameters follow the same pattern.

4.3 Posterior Predictive

It follows from the posterior in Equation D.4, that the posterior predictive $\Pr(\hat{t}_i \mid \mathbf{p}_i \tilde{T}_i;\theta)$—which we seek to optimize in the objective function in Equation D.1—follows a student's t-distribution $t_{\nu_i}(\hat{t}_i \mid \hat{\mu}_i, \hat{\sigma}_i)$ with $\nu_i = 2\alpha_{i,m}$ degrees of freedom, location $\hat{\mu}_i = \mu_{i,m}$, and scale $\hat{\sigma}_i = \sqrt{\frac{\beta_{i,m}(\kappa_{i,m}+1)}{\alpha_{i,m}\kappa_{i,m}}}$ [45], and with probability density function

$$h(t_i \quad | \quad \nu_i, \hat{\mu}_i, \hat{\sigma}_i) \quad = \quad \frac{\Gamma(\frac{\nu_i+1}{2})}{\Gamma(\frac{\nu_i}{2})\sqrt{\nu_i \text{ß}}\hat{\sigma}_i}\left(1+\frac{1}{\nu_i}\left(\frac{t_i-\hat{\mu}_i}{\hat{\sigma}_i}\right)^2\right)^{-\frac{\nu_i+1}{2}}. \tag{D.6}$$

4.4 A Prior Function Layer

To illustrate how to use G-UniTE with neural networks, we present a prior function layer in Algorithm 5 that outputs the prior hyperparameters in G-UniTE. The prior function layer is intended to be used as the final layer of a neural network s.t. the neural network models the prior function $f(\mathbf{p}_i, \boldsymbol{\tau}_i; \psi)$ in Figure 2, where ψ are neural network weights.

Algorithm 5 Forward Propagation through the Prior Function Layer

```
1: function PriorFunctionLayer($\mathbf{x}_i$)
2:     $x_i \leftarrow h(\mathbf{p}_i, \boldsymbol{\tau}_i; \psi_h)$
3:     $\mathbf{h}_1 \leftarrow \mathbf{W} \cdot \mathbf{p}_i$
4:     Let: $\mathbf{h}_1 = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} & h_{1,4} \end{bmatrix}$
5:     $\mu_{i,0} \leftarrow h_{1,1}$
6:     $\kappa_{i,0} \leftarrow \mathrm{ELU}_a(h_{1,2}) + a + \epsilon$
7:     $\alpha_{i,0} \leftarrow |h_{1,3}| + \epsilon$
8:     $\beta_{i,0} \leftarrow |h_{1,4}| + \epsilon$
9:     return $\begin{bmatrix} \mu_{i,0} & \kappa_{i,0} & \alpha_{i,0} & \beta_{i,0} \end{bmatrix}$
```

The prior function layer in Algorithm 5 takes as input a feature vector $\mathbf{x}_i$. In the context of neural networks, $\mathbf{x}_i$ may be the result of a function h s.t. $h(\mathbf{p}_i, \boldsymbol{\tau}_i) = \mathbf{x}_i$ where h represents forward propagation of vectors $\mathbf{p}_i$ and $\boldsymbol{\tau}_i$ through multiple layers. In lines 3–4, $\mathbf{x}_i$ is projected to a four-dimensional vector $\mathbf{h}_1$, one for each of the prior hyperparameters, using a learnable weight matrix $\mathbf{W}$. Recall from Section 4.1 that the prior hyperparameters are constrained s.t. $\kappa_{i,0} > 0$, $\alpha_0 > 0$, and $\beta_0 > 0$. These constraints are enforced in lines 6–8. The values $h_{1,3}$ and $h_{1,4}$ are interpreted as the prior hyperparameters $\alpha_{i,0}$ and $\beta_{i,0}$ and are constrained by taking their absolute values and adding a small non-zero positive constant ϵ to ensure that they are greater than zero. We chose this way of enforcing non-negativity due to its simplicity.

We initially constrained the value $h_{1,2}$, interpreted as the prior hyperparameter $\kappa_{i,0}$, in the same way as $h_{1,3}$ and $h_{1,4}$. However, as discussed in Section 4.2, $\kappa_{i,0}$ represents the confidence in the prior, i.e., the output of the function-fitting component. Experiments showed that, since the function-fitting component performs poorly in the initial stages of training, the value of $\kappa_{i,0}$ will be very low. To alleviate this problem, we use the expression in line 6 of Algorithm 5 instead, which makes use of the Exponential Linear Unit (ELU) [46] function

$$\mathrm{ELU}_a(x) = \begin{cases} x & x > 0 \\ a(\mathrm{e}^x - 1) & x \leq 0, \end{cases} \tag{D.7}$$

where $a > 0$.

Because Equation D.7 has a minimum value of $-a$, we can enforce non-negativity of $\kappa_{i,0}$ by adding a and ϵ, as shown in line 6 of Algorithm 5. This expression makes the value of $\kappa_{i,0}$ less sensitive to changes that decrease its value, thus discouraging decreases of $\kappa_{i,0}$ during early stages of training that needs to be corrected in later stages. Hyperparameter a regulates this effect, s.t. the effect is inversely proportional to a. In addition, the value of $h_{1,2}$ is initially very close to zero in a neural network setting. Using $\kappa_{i,0} = |h_{1,2}| + \epsilon$ would therefore result in a $\kappa_{i,0}$ value close to zero indicating, an unreasonably low confidence in the model. The constraint measure used in line 6 in Algorithm 5 instead ensures that the initial value of $\kappa_{i,0}$ is close to a. Preliminary experiments showed performance improvements when enforcing non-negativity of $\kappa_{0,i}$ in this way, as opposed, taking the absolute value and adding a small constant, but they showed improvements when non-negativity of $\alpha_{i,0}$ and $\beta_{i,0}$ were enforced in the same way.

5 Empirical Study

We evaluate UniTE on the task of trajectory travel time estimation. In particular, we are interested in evaluating UniTE's capability for improving estimation of the travel speed distributions of road segments traversed during a trip over function-fitting and aggregation-based approaches, but also UniTE's capability for improving point estimates of travel times. In addition, we investigate the behavior of UniTE under varying degrees of data availability and different choices of parameters.

5.1 Dataset

For our experiments, we use a dataset of 336 253 trajectories from Aalborg Municipality in Denmark that occurs between January 1st 2012 and December 31st 2014 [47]. The trajectories have been map-matched to the road network of Aalborg Municipality extracted from OSM [48] with 16 294 intersections and 35 947 road segments. See Section 2.1 and Section 2.1 for details on road network trajectory modeling, respectively. See [47] for details on the trajectory data and map-matching process.

We use the 148 746 trajectories from the period January 1st 2012 to June 31st 2013 for training and set aside the 72 693 trajectories from July 1st 2013 to December 31st 2013 for validation. We use the remaining 114 028 trajectories from January 1st 2014 to December 31st 2014 for testing. To characterize each road segment in a trajectory, we use a set of 16 features derived from OSM data and data from the Danish business authority [35]. These road segment feature representations are sparse, containing information about just four attributes: road segment length, road segment category (e.g., motorway), and the kind of zone (city, rural, or summer cottage) the source and target intersections of the road segment are in. The sparsity in the feature representation makes function-fitting difficult [34, 35]. In addition, 19 510 (54%) of the road segments are annotated with a speed limit derived from OSM and municipal data [35]. For further details, see [35].

5.2 Objective Function

We optimize for travel speed distribution modeling performance using the per-trajectory mean Negative Log-Likelihood (NLL). The NLL of a trajectory *TR* is

$$\mathrm{NLL}(TR) = \sum_{(e_i,\tau_i,t_i)\in TR, t_i\neq\varnothing} \mathrm{sNLL}(e_i,\tau_i,t_i) \tag{D.8}$$

where $\mathrm{sNLL}(e_i,\tau_i,t_i) = -\ln \Pr(t_i \mid \theta)$ and θ are model parameters. In the case of UniTE, $\Pr(t_i \mid \theta)$ is the Gaussian posterior predictive (see Equation D.6). The NLL directly measures the likelihood of a trajectory occurring by considering how well an algorithm models the travel speed distributions of its constituent road segments. A good model achieves a high likelihood across all trajectories, resulting in a low NLL score.

5.3 Algorithms

In our empirical study, we combine a function-fitting baseline and an aggregation-based baseline in a unified approach using the UniTE framework and compare their separate performance with their unified performance.

The output of all algorithms is the density function $\Pr(\hat{t}_i = t_i \mid \theta)$ used in Equation D.8 where θ are model parameters specific to each algorithm. All algorithms are implemented in Python 3[1] and trained using the MXNet deep learning framework[2]. We have made the implementation of all algorithms publicly available[3].

[1] https://www.python.org/

[2] `https://mxnet.incubator.apache.org`

[3] To be released upon acceptance.

AGG

Existing aggregation-based approaches [36–39] do not differ in how they function w.r.t. to our empirical study. We therefore use the AGG baseline to represent these approaches collectively in the study.

Firstly, when estimating the travel speed distribution of a road segment e at time τ, these approaches aggregate historical records from trajectories where the road segment is traversed at a similar time within some interval of size δ. Secondly, rather than modeling uncertainty about the hyperparameters of the distribution model like UniTE, they set a threshold k for the minimal number of historical records considered sufficient. If the number of historical records is insufficient, an estimate is derived from the speed limit [36–39].

We represent existing aggregation-based approaches using a single baseline algorithm AGG with the two features of existing aggregation-based approaches. For a fair comparison with G-UniTE, AGG also models travel speed distributions as Gaussian distributions rather than histograms [36–38] or a mean value [39]. See Section A for a detailed description of the AGG baseline.

Record Selection AGG's performance is strongly dependent on the record selection strategy used. In general, aggregation-based approaches must balance *record relevance* with *record availability* The selection strategy used by AGG considers two kinds of relevance: contextual relevance and temporal relevance.

The contextual relevance hyperparameter c is an integer that adjusts the contextual relevance where the context is the c preceding and succeeding road segments in the trajectory from which a historical records originates. Only historical travel speed records from the training trajectories with the same context are selected for aggregation. Thus, increasing c increases contextual relevance.

The temporal relevance parameter δ is given in some unit of time and adjusts how inclusive the record selection strategy is w.r.t. historical records from different times of week. For instance, when estimating for time τ, contextually-relevant historical records occurring in the time interval $[\tau_i - \frac{\delta}{2}; \tau_i + \frac{\delta}{2}]$ are selected.

The record selection algorithm is described fully in Section A.2.

GRU

State-of-the-art function-fitting methods for travel time estimation of routes [19, 21–23, 26–28, 30, 31, 36, 38]are not compatible with our map-matched vehicle trajectory data. In addition, they have been designed for point estimates of travel time, rather than distribution estimates. For these reasons, we instead use a simple recurrent neural network, *GRU*, as the function-fitting approach in our study The GRU model features a GRU cell [49] with a skip connection and is thus a recurrent neural network like many state-of-the-art methods. GRU uses the prior function layer described in Algorithm 5 as the final layer. A full specification of GRU can be found in Section B.

Unlike AGG, recurrent models can model correlations in segment travel speeds within a trajectory. For instance, if a vehicle drives onto a motorway segment 'A' from a motorway approach 'B' with a lower speed limit than the motorway, some time is spent on acceleration to cruising speed. On the other hand, if the vehicle drove onto motorway segment 'A' from an adjacent motorway segment 'C' less time, if any, is spent on acceleration assuming similar traffic conditions.

UniTE

We unify the AGG and GRU baselines using the UniTE framework following the instructions in Section 3.3. In brief, we use the function-fitting approach, GRU, to estimate the prior and the record selection strategy of the aggregation-based approach, AGG, to compute the posterior. Although UniTE is not guaranteed to outperform AGG or GRU on average, we expect it to do so if it is not overfit, since, as discussed in Section 3.3, UniTE can always produce a model that is no worse than its aggregation-based or function-fitting component. See Section C for a comparison of reported errors in the literature.

As mentioned in Section 3.2, we train the generative UniTE model using a discriminative objective. To evaluate the this decision, we consider both a discriminative and a generative variant of UniTE.

UniTE-DIS is UniTE as described in Section 3 where the posterior predictive is optimized directly during training. This is considered an *end-to-end* approach from a machine learning perspective. UniTE-GEN instead operates in two step. First, UniTE-GEN outputs only the prior predictive at training time and then computes the posterior predictive only at test time.

A benefit of UniTE-GEN is that it can be applied to already training function-fitting approaches. In our empirical study, we always reuse a GRU model for UniTE-GEN in a one-to-one fashion. That is, whenever we train and evaluate a GRU model, we also evaluate the corresponding UniTE-GEN model.

5.4 Evaluation Metrics

We use NLL (see Equation D.8) to evaluate travel speed distribution estimation of each algorithm in our study. In addition, we evaluate each algorithm's travel time point estimation performance using Mean Absolute Error (MAE), a commonly used measure for this purpose. Finally, to make our results more easily comparable to those of other papers using different methods and datasets, we also measure the Mean Absolute Percentage Error (MAPE) of the algorithms used in our empirical study, another commonly used measure in the traffic travel time and speed estimation literature.

We compute a travel time point estimate for a trajectory following route $p = (e_1, \dots, e_n)$ as $\sum_{i=1}^{n} \frac{l_i}{\mathbb{E}[d_i]} = \sum_{i=1}^{q} \frac{l_i}{\hat{\mu}_i}$ where l_i is the length of road segment e_i and $\hat{\mu}_i = \mu_{i,m}$ is the expected travel speed computed using Equation D.5. Recall that $m = 0$ for GRU.

The MAE is the mean absolute error of the estimated and actual travel time point estimate. For a trajectory TR, the absolute error is $\text{AE} = |\hat{y}_{TR} - y_{TR}|$ where $\hat{y}_{TR}$ and y_{TR} is the travel time point estimate and ground truth, respectively, of trajectory T.

The MAPE is the mean absolute percentage error of the estimated and actual travel time point estimate. For a trajectory TR, the absolute percentage error is $\text{APE} = \frac{|\hat{y}_{TR} - y_{TR}|}{y_{TR}}$.

5.5 Training and Hyperparameter Selection

The GRU and UniTE-DIS models are trained by to minimizing the NLL in Equation D.8 across all trajectories in the training using the ADAM optimizer [50]. Each GRU model in our study is reused in a UniTE-GEN model as the function-fitting component.

Based on preliminary experiments, we train each GRU and UniTE-DIS model for 10 epochs with a batch size of 128 trajectories. The training trajectories are divided uniformly at random into 1163 batches and are reused across all epochs. The batches are shuffled before each epoch s.t. the are in random order. For both UniTE-DIS and GRU, we selected the learning rate λ by training a GRU model using for each learning rate $\lambda \in \{0.1, 0.01, 0.001, 0.0001\}$. We use the learning rate $\lambda = 0.001$ with the best validation NLL.

For AGG, we selected aggregation threshold parameter k, contextual relevance parameter c, and the temporal relevance parameters δ using a grid search over values $k \in \{1, 2, 4, 8\}$, $c \in \{0, 1, 2, 4\}$ and $\delta \in \{15, 30, 60, 120\}$. We selected the hyperparameter configuration $k = 1$, $c = 0$, and $\delta = 120$ with the best validation NLL.

The parameter a used in the prior function layer (see Algorithm 5) serves a similar role as the aggregation threshold k used in AGG. We therefore set $a = 1$ based on the best value $k = 1$ for AGG. We choose hyperparameters c and δ using a grid search over the same values as for AGG. For UniTE-DIS the best hyperparameters are $c = 1$ and $\delta = 120$, and $c = 4$ and $\delta = 15$ for UniTE-GEN.

Table 1: Algorithm performance on the test trajectories.

Algorithm	*NLL*	*MAE (s)*	*MAPE (%)*
UniTE-DIS	**26.83 ± 0.52**	**75.13 ± 0.70**	**17.48 ± 0.15**
UniTE-GEN	42.99 ± 7.84	83.35 ± 1.66	20.86 ± 0.42
GRU	44.47 ± 1.00	97.38 ± 2.77	24.82 ± 0.81
AGG	75.51 ± 0.00	77.09 ± 0.00	18.53 ± 0.00

5.6 Performance Evaluation

We repeat our experiments ten times for each algorithm and report their mean performance with standard deviations in Table 1. However, one of the GRU models did not finish properly. The numbers for GRU and UniTE-GEN in Table 1 are therefore based on nine runs. Since the AGG baseline is deterministic, no standard deviations are associated with its performance figures.

Distribution Modeling As shown in Table 1, UniTE-DIS is the best-performing algorithm on average for both travel speed distribution modeling and travel time point estimation. On average, UniTE-DIS outperforms the GRU and AGG baselines by 39.68% and 64.48%, respectively, on distribution modeling performance in terms of NLL. UniTE-GEN also outperforms the baselines on distribution modeling, although not as substantially as UniTE-DIS, with improvements of 43.07% and 3.33% over AGG and GRU, respectively. In addition, the results vary substantially between runs with a standard deviation of 7.48. Given these variations in NLL in combination with the low number of runs of UniTE-GEN, it is unclear whether UniTE-GEN outperforms GRU in general.

A detailed analysis of the results suggests that UniTE-GEN implicitly sacrifices estimation accuracy of the distribution mean for estimation accuracy of the distribution variance, a pattern not found in UniTE-DIS or the GRU baseline. Specifically, when comparing any two runs of UniTE-DIS and the GRU baseline, the run with the best estimation of the distribution typically also has the distribution variance estimate. In addition, the GRU model that provides the prior hyperparameters has been pre-trained on the data and therefore also to estimate travel speed distribution. However, as a consequence of the rules for computing the posterior hyperparameters in Equation D.5, the degrees of freedom of the estimated *t*-distribution of the posterior predictive increases with the number of historical records used. This results in a reduced spread of the distribution, particularly for road segments with relatively few historical records that do not capture the travel speed distribution.

Travel Time Point Estimation Interestingly, the ranking of the algorithms w.r.t. travel time point estimation is different from the ranking w.r.t. distribution modeling, as shown in Table 1. The differences between the algorithms are also smaller, with the GRU baseline being a notable exception. UniTE-DIS remains best with average performance increases of 2.54%, 9.86, and 22.85% over AGG, UniTE-GEN, and GRU, respectively, in terms of MAE. Unlike for the task of distribution modeling, UniTE-GEN provides a substantial performance improvement of 14.40% over the GRU baseline for travel time point estimation, with a smaller standard deviation, i.e., 1.66 vs. 2.77 for MAE.

Summary UniTE-DIS performs 39.68–64.48% and 2.54–22.85% better on distribution modeling and travel time point estimation, respectively, than the function-fitting and aggregation-based baselines. In addition, UniTE-DIS outperformed UniTE-GEN by 9.86–37.60% across all measures and does not suffer from the mean-variance estimation accuracy trade-off we observed in the UniTE-GEN model, leading to large variances in distribution modeling performance. These findings show that, while perhaps unconventional, training UniTE models to optimize a discriminative objective rather than a conventional generative objective, is worthwhile.

5.7 The Generalizability-Accuracy Trade-Off

One of our primary goals for UniTE is that UniTE models are flexible s.t. they can exploit the generalizability of the function-fitting component in data-sparse situations and can exploit the accuracy of aggregation-based methods in data-abundant situations. To investigate this capability, we collect the sNLL of different road segments during evaluation of the test trajectories. Then, we group them by the number of historical records available according to the ground-truth arrival times at the segments and compute the mean sNLL for each of these groups using the least restrictive record selection strategy (AGG's). The number of historical records used by each algorithm may differ from this number, depending on the restrictiveness of the record selection strategy and the accuracy of the expected arrival time at the road segments, but they are very strongly correlated. The relationship between mean sNLL and the number of historical records available is shown in Figure 3.

Analysis Figure 3 shows that AGG has substantially worse performance than the other algorithms when few historical records are available, but that it overtakes GRU when 8 historical records available. Until about 35 historical records are available, UniTE-DIS has the best performance, but from this point on, AGG has similar performance.

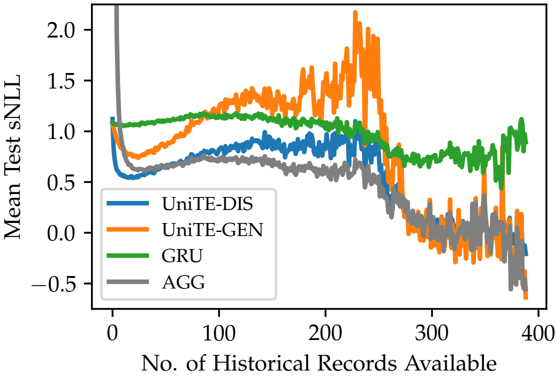


Fig. 3: The relationship between mean sNLL for road segment with different number of historical records available during evaluation on the test trajectories.

UniTE-DIS and AGG maintain quite similar performance when 80 to 250 historical records are available. In this interval, the historical records have high quality, and there are sufficiently many. The AGG baseline therefore overtakes UniTE-DIS and UniTE-GEN in terms of performance because it relies solely on the data and does not make use of a prior. In addition, the record selection strategies of UniTE-DIS and UniTE-GEN are more restrictive, causing them to use, respectively, 34% and 91% less data on average than does AGG. The adverse effect of using a prior in this interval is particularly strong for UniTE-GEN. We expect this is (a) because it has the by far most restrictive record selection strategy and (b) because it expresses much higher certainty in the prior than UniTE-DIS does. When more than 250 historical records are available, UniTE-DIS and UniTE-GEN achieve similar performance to AGG since the influence of the prior becomes less significant.

Summary Both UniTE variants exhibit better generalizability than AGG when few historical records are available and achieve similar accuracy when many historical records are available, where UniTE-DIS achieves superior generalizability and accuracy compared to UniTE-GEN. Between these two extremes, AGG is superior to the UniTE variants due to the availability of sufficient high-quality data. These data conditions are particularly favorable for AGG since it does not make use of a prior that may be inaccurate.

5.8 Regularizing Properties

An important property of the UniTE framework is the implicit regularization w.r.t. data imbalances that results from the definition of the posterior. In

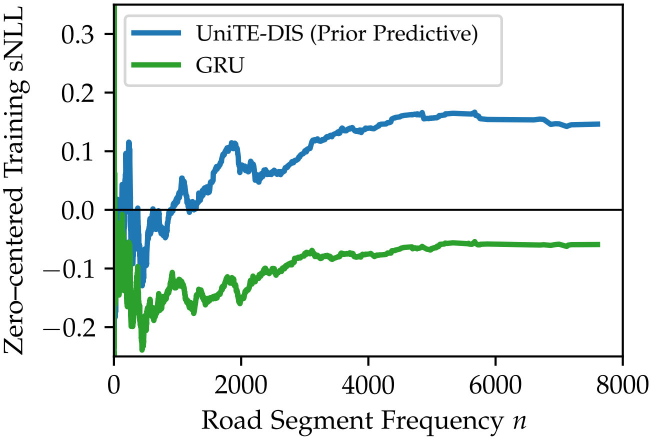


Fig. 4: Moving average (sample window size **250**) of the zero-centered training sNLL of the prior predictive of UniTE-DIS and GRU at different road segment frequencies.

particular, the posterior (cf. Equation D.5) implicitly regularizes the prior function since the influence of the prior function is inversely proportional to the number of historical records. When training UniTE-DIS, we expect that this implicit regularization encourages the GRU architecture used as the prior function to output prior travel speed distributions that are accurate when only few historical records are available. However, we do not expect this effect in GRU (or, equivalently, in UniTE-GEN) since historical records are not used at training time.

To study the regularization properties, we compare the segment-wise mean NLL, sNLL, of UniTE-DIS and GRU on the training set. Let e be a road segment that occurs n times in the set of training trajectories. Then, the sNLL of e is $\frac{1}{n}\sum_{i=1}^{n} - \ln p_i$, where $p_i = \Pr(t_i \mid \mathbf{e}, \tau_i, \tilde{T}_i = \emptyset; \theta)$ is the value of the density function of the prior predictive at the ith occurrence. Here, τ_i is the time of the occurrence, and t_i is the ground truth travel speed.

Figure 4 plots the sNLL of all road segments as a function of their frequency. For ease of comparison, we have centered the values around 0 and use a moving average with a sample window size of 250. Thus, values above 0 indicate below average performance and values below 0 indicates above than average performance within the sample window.

As expected, the prior predictive of UniTE-DIS favors low-frequency road segments more than GRU and GRU favors high-frequency road segments more than UniTE-DIS. Although not visible in Figure 4, the point of diversion occurs at a road segment frequency of $n = 38$ corresponding to the 56th percentile. This discrepancy continue to increase as the road segment frequency increases. These findings suggest that the implicit regularization of the UniTE framework contributes to the superior performance, shown in Table 1, of UniTE-DIS.

5.9 Data Efficiency

We investigate how the performance of the algorithms changes depending on the training data available by giving them part of the training data, while keeping the number of iterations (i.e., backpropagations) constant. For the sake of brevity, we show only the results in terms of test NLL in Figure 5, but the patterns when using MAE and MAPE are similar.

AGG and GRU As shown in Figure 5, the performance of AGG is highly dependent on the size of the training set, whereas GRU is comparatively good at generalizing when using few training trajectories. As discussed in Section 1, function-fitting approaches such as GRU are good at generalization, and aggregation-based approaches such as AGG are not. The results are therefore as expected. However, it is notable that the performance of GRU is near-constant (within six standard deviations) and does not improve as more data becomes available. We expect the primary cause to be the lack of high quality features available: using four attributes represented as 16 features means that the feature space can be observed almost completely through few trajectories.

UniTE-GEN Figure 5 shows that UniTE-GEN tends to nearly-match or outperform it's pre-trained GRU component. The results also suggest that UniTE-GEN tends to scale better with training data availability. However, the value of the UniTE framework when used in a generative manner is highly dependent on the pre-trained GRU model it uses. As shown in the figure, this dependence results in a large performance variance that is consistent with the results in Table 1. However, when measuring travel time point estimation using MAE or MAPE (not shown), UniTE-GEN is strictly superior to its pre-trained function-fitting GRU component for all data set sizes and the performance difference increases proportionally to data set size.

UniTE-DIS UniTE-DIS outperforms the other algorithms for all data set sizes. Unlike GRU, UniTE-DIS scales with data availability, although not as aggressively as AGG. The results show that UniTE-DIS has high data efficiency and can substantially outperform a purely function-fitting and a purely aggregation-based approach even at very small data set sizes. For instance, when using ca. 10% of the training trajectories, UniTE-DIS outperforms AGG and GRU by 728% and 20%, respectively.

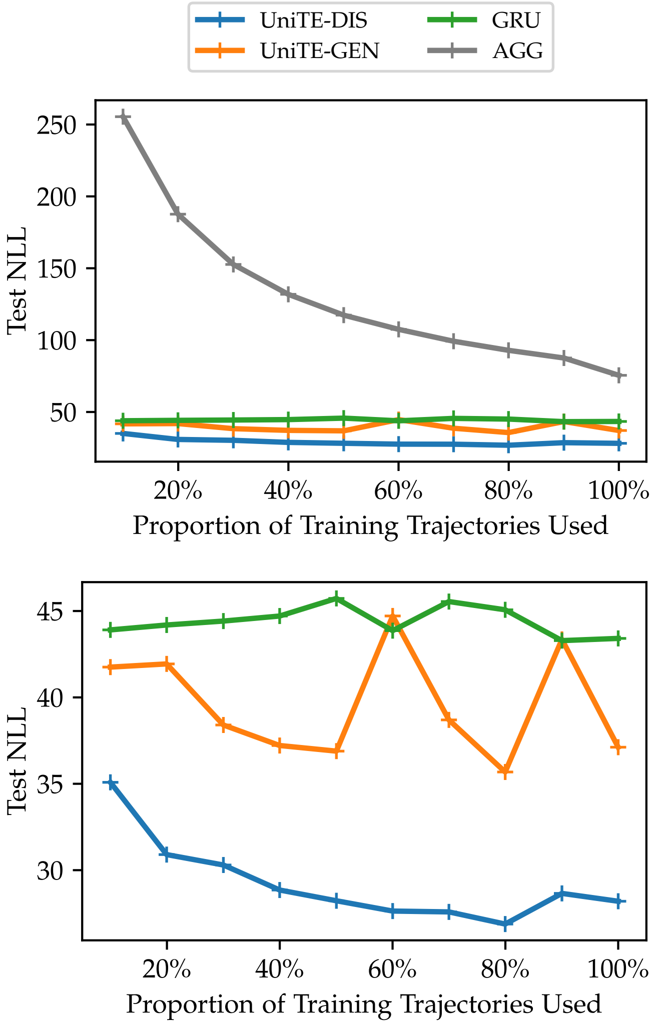


Fig. 5: Algorithm travel speed distribution modeling performance for different data subsets.

Figure 5 suggests that the performance of UniTE-DIS deteriorates beyond 80% of the training trajectories. Given that the differences are small, we expect that this is due to the stochastic nature of the training process, but it may also be due to differences in the distributions of the training and test sets. If the latter is the case, regularization techniques may be used during training to enhance generalizability. However, since the performance differences are within six standard deviations, we cannot conclude which is the case.

Summary UniTE-DIS exhibits superior data efficiency compared to GRU and AGG and achieves superior performance for all data set sizes considered in our study and achieves superior performance at all data set sizes considered in our study. And Unlike the AGG baseline, UniTE-DIS exhibits good generalizability for small data set sizes. And unlike the GRU baseline, the performance of UniTE-DIS improves proportionally to size of the data set. UniTE-GEN exhibits the same behavior as UniTE-DIS when measuring travel time point estimation performance, albeit with strictly worse performance at all data set sizes. However, when measuring distribution modeling performance, the potential performance increase of using UniTE-GEN on a pre-trained GRU model is highly dependent on the particular GRU model.

5.10 Record Selection Strategies

The value of the UniTE framework depends on the record selection strategy. In particular, there is a trade-off between data availability and data quality. If the historical records are irrelevant, the posterior predictive may perform worse than the prior predictive. However, if no historical records are available, UniTE offers no benefits (but also no drawbacks). In addition, we hypothesize that optimal record selection strategies for purely aggregation-based approaches differs from optimal selection strategies for UniTE. We therefore investigate how different values of the temporal relevance hyperparameter δ and the contextual relevance hyperparameter c influence AGG, as well as UniTE-DIS and UniTe-GEN. Recall that large c values indicates high relevance and high δ indicates low relevance.

Analysis Figure 6 shows the validation NLL found during hyperparameter selection to select the best combination of c and δ values for AGG, UniTE-DIS, and UniTE-GEN. As shown in the figure, the optimal values differ across the algorithms. In addition, they follow quite different patterns.

AGG benefits from high data availability even at the expense of data relevance. This is not surprising given the inaccurate heuristic used to estimate the travel speed distributions when too few records are available. UniTE-GEN follows the opposite pattern and prefers high data relevance. This is

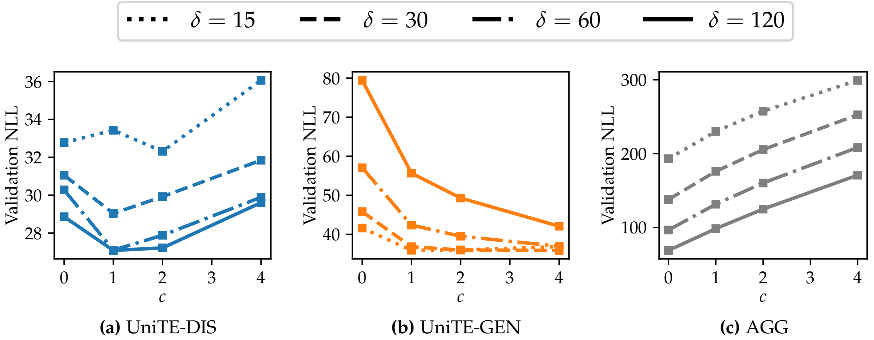


Fig. 6: The distribution modeling performance on the validation set of (a) UniTE-DIS, (b) UniTE-GEN, and (c) AGG for different values of parameters c and δ that regulate contextual and temporal relevance of the retrieved historical records.

likely caused by the variance-reducing effects of UniTE-GEN discussed in Section 5.6, leading to too narrow travel speed distributions. If few records are available, this effect is not as pronounced. Finally, UniTE-DIS is somewhere in-between, preferring a moderate contextual relevance with $c = 1$. Temporal relevance is less important, with $\delta = 60$ and $\delta = 120$ yielding nearly equal performance, particularly when $c = 1$.

Summary We find that optimal the record selection strategies differ across the UniTE and AGG. In particular, our results suggest that optimal strategies for UniTE are more selective than those for aggregation-based approaches. We attribute this to the difference in the quality of the 'default' mechanism used to find travel speed distributions when few or no historical records are available.

6 Related Work

As discussed in Section 1, approaches for travel time and speed estimation can be categorised broadly as either function-fitting [3–33, 35] or aggregation-based [36–39] approaches.

Function-fitting approaches differ primarily in how they structure the function they are fitting, and aggregation-based approaches primarily differ in how they select or construct records for aggregation. The details of these approaches are orthogonal to the novelty of the UniTE framework. The UniTE framework can be used in conjunction with existing function-fitting as well as aggregation-based approaches, where existing function-fitting approaches can be used as the function-fitting component in UniTE and existing aggregation-based methods can be used for record selection and construction in UniTE. To the best of our knowledge, UniTE is the first approach that combines function-fitting and aggregation-based approaches in a single cohesive framework. We expect that only minor modifications to the output and objective of a function-fitting approach is necessary, and that no changes to the record selection strategies of aggregation-based approaches are necessary.

Approaches to travel time and speed estimation can further be categorised as segment-based [3, 4, 15, 17, 20, 24, 29, 32, 33, 35, 37], route-based [19, 21–23, 26–28, 30, 31, 36, 38], origin-destination based [7, 18, 25, 39], or station-based [5, 6, 8–14, 16], meaning that they target estimation for segments, routes, origin-destination pairs, and traffic measuring stations, respectively. Traffic measuring stations typically represent loop detectors in traffic forecasting applications.

In this paper, we have adopted a segment-based and a route-based perspective, but the UniTE framework only requires that the type of data instance, be it a segment, route, origin-destination pair, or a measuring station, is represented as a feature vector. The framework is thus equally compatible with origin-destination-based and station-based approaches, given that appropriate methods of feature vector construction and record selection are available.

Early Version of UniTE An early version of the UniTE framework has been presented in the Master's thesis of [51] . The present version is generalizes the earlier version s.t. the earlier version is a concrete realization of UniTE similar to G-UniTE. Like G-UniTE, the early version assumes that travel speed distributions are Gaussian, but only model uncertainty about the distribution means, not the variances, and provides only point estimates of travel time or travel speed. In addition, the early version considers only a post-training computation of the posterior similar to the UniTE-GEN algorithm used in our empirical study. As a result, the early version does not inherently regularize the function-fitting component to account for data imbalances. In addition, the present version of UniTE is not restricted to discrete time like the earlier version, but also supports continuous time. Finally, this work performs presents a more extensive evaluation with an in-depth analysis of the data scalability, data efficiency, and the trade-off between data quantity and data quality when selecting historical records.

7 Conclusions and Future Work

We have presented UniTE, a novel framework that provides a Unifying Approach to Travel time and speed Estimation. UniTE unifies function-fitting and aggregation-based approaches to travel time and speed estimation to leverage the generalizability of function-fitting approaches with the accuracy of aggregation-based approaches. By virtue of being a Bayesian framework, UniTE is able to switch smoothly between its constituent function-fitting and aggregation-based components depending on data availability.

In our empirical study, we found that UniTE can improve the accuracies of travel speed distribution and travel time estimation by 40–64% and 3–23%, respectively, compared to using function fitting or aggregation alone. These improvements result from the superior generalizability relative to both the function-fitting approach and the aggregation-based approach in our study, while maintaining superior or similar accuracy relative to the aggregation-based approach across all data availability scenarios in our dataset. We used the comparatively simple GRU baseline as a substitute for the incompatible state-of-the-art route-based function-fitting approaches to travel time estimation in our study. As discussed in Section 3.3, there always exists an UniTE model that can achieve the same performance as either its function-fitting or aggregation-based component within arbitrary precision. In addition, reported errors for state-of-the-art approaches on different datasets are roughly half of GRUs error, making them quite comparable in terms of estimation performance. We therefore still expect such approaches to benefit from integration with an aggregation-based approach using UniTE although the estimation performance gains are likely smaller the more accurate the function-fitting approach is independently. See Section C for a more comprehensive discussion.

UniTE has a number of other benefits in addition to estimation performance improvements. First, the framework implicitly regularizes its function-fitting component to handle issues of data imbalance and reduce model bias due to its Bayesian nature. Second, UniTE models are less reliant on the structural quality of both the mapping function, i.e., the neural network architecture in our empirical study, and input feature vector representations since they also use aggregation during estimation. This property of UniTE can reduce the typically substantial resources required for feature engineering and neural network architecture design when using neural networks for function-fitting.

Future directions for UniTE include exploring more complex models of travel time and speed distributions. This may necessitate more sophisticated techniques than the conjugate Bayesian analysis [45] used in G-UniTE, e.g., variational inference techniques [52]. In addition, investigating further synergistic effects of function-fitting and aggregation within the UniTE framework is of interest. For instance, in our empirical study, the internal state of the recurrent Gated Recurrent Unit (GRU) cell used in the UniTE models is unaffected by the computation of the posterior. This makes it more difficult for GRU cells to leverage correlations in travel speed between adjacent road segments in a route for better estimation accuracy.

References

[1] Z. Liu, L. Chen, and Y. Tong, "Realtime Traffic Speed Estimation with Sparse Crowdsourced Data," in *Proc. of ICDE*, 2018, pp. 329–340.

[2] F. Barth, S. Funke, T. S. Jepsen, and C. Proissl, "Scalable Unsupervised Multi-Criteria Trajectory Segmentation and Driving Preference Mining," in *Proc. of BigSpatial*, 2020, pp. 1–10.

[3] B. Yang, M. Kaul, and C. S. Jensen, "Using incomplete information for complete weight annotation of road networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 5, pp. 1267–1279, 2013.

[4] J. Zheng and L. M. Ni, "Time-dependent trajectory regression on road networks via multi-task learning," in *Proc. of AAAI*, 2013, pp. 1048–1055.

[5] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *Proc. of IJCAI*, 2017, pp. 3634–3640.

[6] Z. Cui, K. Henrickson, R. Ke, and Y. Wang, "Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4883–4894, 2020.

[7] H. Yuan, G. Li, Z. Bao, and L. Feng, "Effective travel time estimation: When historical trajectories over road networks matter," in *Proc. of SIGMOD*, 2020, p. 2135–2149.

[8] C. Wei and J. Sheng, "Spatial-temporal graph attention networks for traffic flow forecasting," in *IOP Conference Series: Earth and Environmental Science*, vol. 587, 2020, paper no. 012065.

[9] H. Lu, D. Huang, Y. Song, D. Jiang, T. Zhou, and J. Qin, "ST-TrafficNet: A spatial-temporal deep learning network for traffic forecasting," *MDPI: Electronics*, vol. 9, 2020, paper no. 1474.

[10] L. Ge, S. Li, Y. Wang, F. Chang, and K. Wu, "Global spatial-temporal graph convolutional network for urban traffic speed prediction," *MDPI: Applied Sciences*, vol. 10, no. 4, 2020, paper no. 1509.

[11] Y. Zhang, T. Cheng, Y. Ren, and K. Xie, "A novel residual graph convolution deep learning model for short-term network-based traffic forecasting," *International Journal of Geographical Information Science*, vol. 34, no. 5, pp. 969–995, 2020.

[12] K. Zhang, F. He, Z. Zhang, X. Lin, and M. Li, "Graph attention temporal convolutional network for traffic speed forecasting on road networks," *Transportmetrica B: Transport Dynamics*, vol. 9, no. 1, pp. 153–171, 2020.

[13] S. Zhang, L. Zhou, X. Chen, L. Zhang, L. Li, and M. Li, "Network-wide traffic speed forecasting: 3d convolutional neural network with ensemble empirical mode decomposition," *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 10, pp. 1132–1147, 2020.

[14] S. Yin, J. Wang, Z. Cui, and Y. Wang, "Attention-enabled network-level traffic speed prediction," in *Proc. of ISC2*, 2020, pp. 1–8.

[15] Y. Lee, H. Jeon, and K. Sohn, "Predicting short-term traffic speed using a deep neural network to accommodate citywide spatio-temporal correlations," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 3, pp. 1435–1448, 2021.

[16] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in *Proc. of AAAI*, vol. 33, 2019, pp. 922–929.

[17] N. Zygouras, N. Panagiotou, Y. Li, D. Gunopulos, and L. Guibas, "HTTE: A Hybrid Technique For Travel Time Estimation In Sparse Data Environments," in *Proc. of SIGSPATIAL*, 2019, p. 99–108.

[18] S. Abbar, R. Stanojevic, and M. Mokbel, "STAD: Spatio-Temporal Adjustment of Traffic-Oblivious Travel-Time Estimation," in *Proc. of MDM*, 2020, pp. 79–88.

[19] L. Fu, J. Li, Z. Lv, Y. Li, and Q. Lin, "Estimation of short-term online taxi travel time based on neural network," in *Proc. of WASA*, 2020, pp. 20–29.

[20] R. Barnes, S. Buthpitiya, J. Cook, A. Fabrikant, A. Tomkins, and F. Xu, "BusTr: Predicting Bus Travel Times from Real-Time Traffic," in *Proc. of SIGKDD*, 2020, p. 3243–3251.

[21] W. Lan, Y. Xu, and B. Zhao, "Travel time estimation without road networks: an urban morphological layout representation approach," in *Proc. of IJCAI*, 2019, pp. 1772–1778.

[22] F. Wu and L. Wu, "DeepETA: A Spatial-Temporal Sequential Neural Network Model for Estimating Time of Arrival in Package Delivery System," in *Proc. of AAAI*, vol. 33, 2019, pp. 774–781.

[23] Y. Shen, J. Hua, C. Jin, and D. Huang, "TCL: Tensor-CNN-LSTM for Travel Time Prediction with Sparse Trajectory Data," in *Proc. of DASFAA*, 2019, pp. 329–333.

[24] J. Hu, C. Guo, B. Yang, and C. S. Jensen, "Stochastic weight completion for road networks using graph convolutional networks," in *Proc. of ICDE*, 2019, pp. 1274–1285.

[25] J. Hu, B. Yang, C. Guo, C. S. Jensen, and H. Xiong, "Stochastic origin-destination matrix forecasting using dual-stage graph convolutional, recurrent neural networks," in *Proc. of ICDE*, 2020, pp. 1417–1428.

[26] T.-y. Fu and W.-C. Lee, "DeepIST: Deep Image-Based Spatio-Temporal Network for Travel Time Estimation," in *Proc. of CIKM*, 2019, p. 69–78.

[27] X. Lin, Y. Wang, X. Xiao, Z. Li, and S. S. Bhowmick, "Path travel time estimation using attribute-related hybrid trajectories network," in *Proc. of CIKM*, 2019, p. 1973–1982.

[28] K. Fu, F. Meng, J. Ye, and Z. Wang, "CompactETA: A Fast Inference System for Travel Time Prediction," in *Proc. of SIGKDD*, 2020, p. 3337–3345.

[29] R. Stanojevic, S. Abbar, and M. Mokbel, "W-Edge: Weighing the Edges of the Road Network," in *Proc. of SIGSPATIAL*, 2018, p. 424–427.

[30] Z. Wang, K. Fu, and J. Ye, "Learning to estimate the travel time," in *Proc. of SIGKDD*, 2018, p. 858–866.

[31] D. Wang, J. Zhang, W. Cao, J. Li, and Y. Zheng, "When will you arrive? estimating travel time based on deep neural networks," in *Proc. of AAAI*, vol. 32, 2018, pp. 2500–2507.

[32] H. Hu, G. Li, Z. Bao, Y. Cui, and J. Feng, "Crowdsourcing-based real-time urban traffic speed estimation: From trends to speeds," in *Proc. of ICDE*, 2016, pp. 883–894.

[33] Y. Wang, Y. Zheng, and Y. Xue, "Travel Time Estimation of a Path Using Sparse Trajectories," in *Proc. of SIGKDD*, 2014, p. 25–34.

[34] T. S. Jepsen, C. S. Jensen, T. D. Nielsen, and K. Torp, "On Network Embedding for Machine Learning on Road Networks: A Case Study on the Danish Road Network," in *Proc. of Big Data*, 2018, pp. 3422–3431.

[35] T. S. Jepsen, C. S. Jensen, and T. D. Nielsen, "Relational Fusion Networks: Graph Convolutional Networks for Road Networks," *IEEE Transactions on Intelligent Transportation Systems*, p. in online early access, 2020.

[36] B. Yang, J. Dai, C. Guo, C. S. Jensen, and J. Hu, "PACE: a PAth-CEntric paradigm for stochastic path finding," *The VLDB Journal*, vol. 27, no. 2, pp. 153–178, 2018.

[37] J. Hu, B. Yang, C. S. Jensen, and Y. Ma, “Enabling time-dependent uncertain eco-weights for road networks,” *GeoInformatica*, vol. 21, no. 1, pp. 57–88, 2017.

[38] J. Dai, B. Yang, C. Guo, C. S. Jensen, and J. Hu, “Path cost distribution estimation using trajectory data,” *Proc. of VLDB*, vol. 10, no. 3, pp. 85–96, 2016.

[39] J. Yuan, Y. Zheng, X. Xie, and G. Sun, “T-drive: Enhancing driving directions with taxi drivers’ intelligence,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 1, pp. 220–232, 2011.

[40] J. Liu, G. P. Ong, and X. Chen, “Graphsage-based traffic speed forecasting for segment network with sparse data,” *IEEE Transactions on Intelligent Transportation Systems*, 2020, in online early access.

[41] L. Wei, Y. Wang, and P. Chen, “A particle filter-based approach for vehicle trajectory reconstruction using sparse probe data,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 5, pp. 2878–2890, 2021.

[42] M.-X. Wang, W.-C. Lee, T.-Y. Fu, and G. Yu, “On Representation Learning for Road Networks,” *ACM Transactions on Intelligent Systems and Technology*, vol. 12, no. 1, 2020.

[43] K. P. Murphy, *Machine learning: a probabilistic perspective.* MIT press, 2012.

[44] J. Salojärvi, K. Puolamäki, and S. Kaski, “On discriminative joint density modeling,” in *Proc. of ECML*, 2005, pp. 341–352.

[45] K. P. Murphy, “Conjugate Bayesian analysis of the Gaussian distribution,” 2007.

[46] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, “Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs) ,” in *Proc. of ICLR*, 2016, p. 14 pp.

[47] O. Andersen, B. B. Krogh, and K. Torp, “An Open-source Based ITS Platform,” in *Proc. of MDM*, vol. 2, 2013, pp. 27–32.

[48] OpenStreetMap contributors, “Planet dump retrieved from https://planet.osm.org,” 2014.

[49] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” in *Proc. of SSST*, 2014.

[50] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *Proc. of ICLR*, 2015, p. 15 pp.

[51] M. Fruensgaard and T. S. Jepsen, "Improving cost estimation models with estimation updates and road2vec: a feature learning framework for road networks," Master's thesis, Aalborg University, 2017.

[52] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

Appendices

A Definition of AGG

The distribution derivation process when no historical records are available is not clear from the literature, but Hu et al. [37] suggests that they use the speed limit as a deterministic (rather than probabilistic) travel speed estimate. A direct application of this approach to our setting results in a Gaussian with the speed limit as the mean and (near-)zero variance. However, such a low variance is unrealistic and severely decreases the travel speed distribution modeling performance of AGG. To achieve a fair comparison, we therefore do the following.

Given a road segment e_i, AGG outputs the mean μ_i and the standard deviation σ_i:

$$
\begin{aligned}
\mu_i &= \begin{cases} M_{\tilde{T}_i} & \text{if } |\tilde{T}_i| \geq k \\ 0.79 \cdot \mathrm{SL}(e_i) & \text{otherwise} \end{cases} \\
\sigma_i &= \begin{cases} S_{\tilde{T}_i} & \text{if } |\tilde{T}_i| \geq k \text{ and } |\tilde{T}_i| > 1 \\ 0.07 \cdot \mu_i & \text{otherwise} \end{cases}
\end{aligned}
\tag{D.9}
$$

Here, $M_{\tilde{T}_i}$ and $S_{\tilde{T}_i}$ are the arithmetic mean and the standard deviation of historical records $\tilde{T}_i$, respectively. The function SL returns the speed limit for its argument road segment e_i.

The factors 0.79 and 0.07 used in Equation D.9 to derive a mean and standard deviation when the number of historical records is insufficient are chosen based on our knowledge of the domain and the dataset, and takes into account that vehicles tend to travel at speeds below the speed limit on urban roads [3], which occur occur in the data set used in our study. As an example, if the speed limit is 50 km/h then drivers drive at around 40 km/h on average, and 99.7% of drivers are expected to drive below the speed limit (as a consequence of the empirical rule). From our experience, this scenario is quite realistic, and using of 79% (rather than 100%) of the speed limit as the distribution mean yielded substantial performance improvements in terms of travel time point estimation in preliminary experiments.

A.1 Speed Limit Derivation

The function invocation $\mathrm{SL}(e_i)$ in Equation D.9 returns the speed limit of road segment e_i if it exists in our dataset. However, as noted in Section 5.1, the dataset used in our study does not contain a speed limit for all road segments. When no speed limit is given, $\mathrm{SL}(e_i)$ instead returns a speed limit derived from road segment e_i's attributes using a OSM heuristic[4].

Since our data is from Denmark, we use the OSM speed limit heuristic for Denmark. It is as follows.

1. If the road category of a road segment is motorway then assign a speed limit of 130.
2. If the road category is trunk then assign a speed limit 80.
3. If the road category is neither motorway or trunk, but the road segment is within a city, then assign a speed limit of 50.
4. Otherwise, assign a speed limit of 80.

A road segment is considered to be in a city if either the source intersection or the target intersection of the road segments is in a city.

A.2 Record Selection

The AGG baseline relies on a record selection strategy to find historical records $\tilde{T}_i$ to compute the sample mean and sample standard deviation used in Equation D.9. The algorithm used for record selection is presented in Algorithm 6.

The algorithm takes as input a set of training trajectories $\mathcal{TR}$, the route p for which the travel time is in the process of being estimated, the road segment $e_i \in p$ for which historical records $\tilde{T}_i$ are currently being collected, and the arrival time τ_i at road segment e_i. In addition, the algorithm takes as input two parameters: an integer contextual relevance parameter, c, and a temporal relevance parameter, δ, in some unit of time. Higher values of c returns fewer, but more relevant historical records. Higher values of δ returns more, but less relevant historical records.

Algorithm 6 constructs the set of historical records $\tilde{T}_i$ as follows. The algorithm scans the set of trajectories in the loop in Lines 5–10 for historical records. For each trajectory, the algorithm scans each traversal in the trajectory for historical records in the loop in Lines 7–10. A historical record refers strictly to the recorded travel time or travel speed of the traversal.

To be selected, a historical record must satisfy the three conditions in Line 9. First, it must be *contextually relevant* s.t. the contexts of road segments e_i and e'_j are identical, i.e., $C_i = C_j$. Here, context refers to the preceding and succeeding road segments of a road segment in a trajectory or a route. Second, the historical record must be *temporally relevant*, i.e., occur at a similar time of week as τ_i, defined by the interval I_i (Line 4). Finally, the historical

[4] `https://wiki.openstreetmap.org/wiki/OSM_tags_for_routing/Maxspeed#Default_speed_limits`

Algorithm 6 Record Selection Algorithm

```
function RecordSelection($\mathcal{TR}$, $p = (e_1, \dots, e_q)$; $e_i$, $\tau_i$, $c$, $\delta$)
    $\tilde{T}_i \leftarrow \emptyset$
    $C_i \leftarrow (e_{i-c}, \dots, e_{i-1}, e_i, e_{i+1}, \dots, e_{i+c})$
    $I_i \leftarrow [\tau_i - \frac{\delta}{2}; \tau_i + \frac{\delta}{2}]$
    for each trajectory $TR \in \mathcal{TR}$ do
        Let: $TR = \big((e'_1, \tau_1, t_1), \dots, (e'_n, \tau_n, t_n)\big)$
        for $j = 1$ to $n$ do
            $C_j \leftarrow (e'_{j-c}, \dots, e'_{j-1}, e'_j, e'_{j+1}, \dots, e'_{j+c})$
            if $C_i = C_j \land \tau_j \in I_i \land t_j \neq \emptyset$ then
                $\tilde{T}_i \leftarrow \tilde{T}_i \cup \{t_j\}$
    return $\tilde{T}_i$
```

record is added to $\tilde{T}_i$ if $t_j \neq \emptyset$, i.e., if the historical record is derived from GPS data and is not created by the map-matching algorithm.

B Definition of GRU

We express GRU in the UniTE framework as the prior function f.

Let $p = (e_1, \dots, e_n)$ be the input route starting at time τ_1. For each road segment e_i in p, GRU computes the following.

$$\begin{aligned} \mathbf{x}_i &= \mathbf{e}_i \oplus \boldsymbol{\tau}_i \\ \mathbf{z}_i &= \mathrm{GRU}(\mathbf{x}_t, \mathbf{z}_{i-1}) \\ \mathbf{h}_i &= \mathbf{z}_i \oplus \mathbf{x}_i \\ f(\mathbf{e}_i, \boldsymbol{\tau}_i; \psi) &= \mathrm{PriorFunctionLayer}(\mathbf{h}_i), \end{aligned} \tag{D.10}$$

where z_0 is a d-dimensional vector of zeros. For $i > 1$, we compute τ_i by incrementing τ_{i-1} by the expected time to traverse road segment e_{i-1}, i.e., we increment τ_{i-1} by $\frac{l_{i-1}}{\mu_{i-1,m}}$ where l_{i-1} is the length of road segment e_i and $\mu_{i-1,m}$ is the expected travel speed when traversing road segment e_i (calculated using Equation D.5). The prior function layer is described in Algorithm 5. During training, we use the time τ_i recorded during the input training trajectory if $\tau_i \neq \emptyset$.

As shown in Equation D.10, GRU takes as input the 32-dimensional vector $\mathbf{x}_i$, a concatenation of the 16-dimensional vector representations of road segment e_i and τ_i. We explain how $\boldsymbol{\tau}_i$ is constructed in Section B.1. Vector $\mathbf{x}_i$ is passed to a GRU cell that outputs a 32-dimensional vector $\mathbf{z}_i$. The GRU cell is recurrent and therefore takes as input vector $\mathbf{z}_{i-1}$, the output of the GRU cell at the previous road segment of the route.

Preliminary experiments indicated that the use of a *skip connection* is beneficial to the GRU baseline, i.e., a connection from an earlier layer to later layer with at least one layer in-between. The skip connection is captured in the computation of $\mathbf{h}_i$ in Equation D.10 where the output of the GRU cell $\mathbf{z}_i$ is concatenated with the input vector $\mathbf{x}_i$. Finally, GRU applies the prior function layer described in Algorithm 5 to vector $\mathbf{h}_i$ to output the prior hyperparameters. Note that $\tilde{T}_i = \emptyset$ when computing the posterior predictive $\Pr(\hat{t}_i = t_i \mid \tilde{T}_i, \theta_i)$ of the GRU baseline, e.g., when computing NLL (cf. Equation D.8) during training or evaluation.

B.1 Representation of Time

The representation of time $\boldsymbol{\tau}_i$ used by the GRU algorithm in Equation D.10 is a 16-dimensional feature vector representation of the time of week, where 8 dimensions are used to represent the time of day and 8 dimensions are used to represent the day of the week. Formally, we learn a time-of-week vector $\boldsymbol{\tau} = \boldsymbol{\tau}_{tod} \oplus \boldsymbol{\tau}_{dow}$ for time τ, where $\boldsymbol{\tau}_{tod} \in \mathbb{R}^8$ represents the time of day, $\boldsymbol{\tau}_{dow} \in \mathbb{R}^8$ represents the day of the week, and $\oplus$ denotes vector concatenation. Preliminary experiments indicated that our results are robust to changes to the dimensionality of this vector representation.

To represent time of day in our experiments, we divide the time of day into 96 15-minute intervals $\mathcal{I} = \{I_1, \ldots, I_{96}\}$ s.t. $I_1 =$ [0:00; 0:15), $I_2 =$ [0:15; 0:30), and so forth. Then, we one-hot encode the time of day τ_{tod} into a 96-dimensional vector $\boldsymbol{\tau}'_{tod} = \begin{bmatrix} \mathbb{1}[\tau_{tod} \in I_1] & \ldots & \mathbb{1}[\tau_{tod} \in I_{96}] \end{bmatrix}$, where $\mathbb{1}$ is the indicator function. Finally, we multiply the one-hot encoding $\boldsymbol{\tau}'_{tod}$ by a trainable matrix $\mathbf{W}_{tod} \in \mathbb{R}^{96 \times 8}$ to achieve the time of day representation $\boldsymbol{\tau}_{tod} = \boldsymbol{\tau}'_{tod}\mathbf{W}_{tod}$ with dimensionality 8.

We represent the day of week in a manner similar to the time of day, but with 7 dimensions in the one-hot encoding $\boldsymbol{\tau}'_{dow}$—one for each day of the week—and multiply $\boldsymbol{\tau}'_{dow}$ by a trainable weight matrix $\mathbf{W}_{dow} \in \mathbb{R}^{7 \times 8}$ to get an 8-dimensional vector representation $\boldsymbol{\tau}_{dow}$ of the day of the week.

C Reported Travel Time Estimation Errors in Other Studies

Utilizing state-of-the-art route-based function-fitting approaches in our empirical study is not possible due to differences in the expected inputs and outputs. Specifically, the input trajectories are available as time-stamped sequences of edges, rather than raw GPS location data, and we are interested in distribution estimates, rather than just point estimates, in our study. These differences make state-of-the-art function-fitting approaches unapplicable for the setting used in our empirical study.

A straight-forward comparison between the state-of-the-art function-fitting approaches and GRU is not possible, but we have gathered reported error rates of different state-of-the-art route-based function-fitting approaches for travel time estimation on different data sets, shown in Table 2. Note, that, unlike GRU, the models in the table are trained using a loss function that optimizes for travel time point estimation directly. Whenever multiple datasets are used in an empirical study, the lowest MAPE across the datasets is included in the table.

Table 2: Reported MAPE of state-of-the-art route-based function-fitting approaches on travel time point estimation.

Algorithm	MAPE
DeepSTTE [11]	10.6%
DeepI2T [21]	15.2%
DeepETA [22]	20.6%
TCL [23]	12.4%
DeepIST [26]	7.8%
AtHy-TNet [27]	10.2%
CompactETA [28]	11.1%
WDR [30]	11.7%
DeepTTE [31]	10.9%

As shown in the Table 2, the reported MAPE in other work is generally between 10% and 15%, which is roughly half the error of GRU. These numbers support our expectation that GRU underperforms compared to state-of-the-art approaches and we expect that the performance increase of UniTE-DIS over GRU will be smaller if UniTE is applied to a function-fitting approach with the same estimation accuracy as state-of-the-art approaches. Furthermore, we expect that performance increases are subject to diminishing returns: the better the function-fitting component, the harder it is to improve upon.

As a rough conservative estimate, we expect that performance increases are between 25–50% of the performance increases achieved by using UniTE-DIS over GRU. In other words, we expect performance increases of 10–20% in terms of NLL, 5–11% in terms of MAE, and 7–15% in terms of MAPE. Note, however, that these estimates assume a scenario where GRU is replaced as the function-fitting component in UniTE-DIS by a function-fitting approach with similar performance as the state-of-the-art approaches in Table 2, but with the same input and output formats as GRU. Performance increases may also surpass the performance increases shown in our empirical study, if using, e.g., a different record selection strategy, a different instance of UniTE that does not assume Gaussian distributions, or UniTE to update the route travel time estimate directly rather than for individual road segments, assuming such data is available.

Paper E

Scalable Unsupervised Multi-Criteria Trajectory Segmentation and Driving Preference Mining

Florian Barth
Universität Stuttgart

Stefan Funke
Universität Stuttgart

Tobias Skovgaard Jepsen
Aalborg University

Claudius Proissl
Universität Stuttgart

This paper is published in the
Proceedings of the 2018 IEEE Internal Conference on Big Data, pp. 3422–3431,
2018.

Abstract

We present analysis techniques for large trajectory data sets that aim to provide a semantic understanding of trajectories reaching beyond them being point sequences in time and space. The presented techniques use a driving preference model w.r.t. road segment traversal costs, e.g., travel time and distance, to analyze and explain trajectories.

In particular, we present trajectory mining techniques that can (a) find interesting points within a trajectory indicating, e.g., a via-point, and (b) recover the driving preferences of a driver based on their chosen trajectory. We evaluate our techniques on the tasks of via-point identification and personalized routing using a data set of more than 1 million vehicle trajectories collected throughout Denmark during a 3-year period. Our techniques can be implemented efficiently and are highly parallelizable, allowing them to scale to millions or billions of trajectories.

Florian Barth, Stefan Funke, Tobias Skovgaard Jepsen, and Claudius Proissl, 'Scalable Unsupervised Multi-Criteria Trajectory Segmentation and Driving Preference Mining.' *In Proceedings of the 9th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, 2020, Article No. 6, pp. 1–10. DOI: 10.1145/3423336.3429348.

The layout has been revised.

1 Introduction

The ubiquity of mobile devices with position tracking capabilities via GPS or localization using WiFi and mobile networks continuously generate vast streams of location data. Such data may be used in a variety of ways. Mobile networks providers and many companies, such as Google or Apple, use the location data of their customers to improve their services, e.g., by monitoring of traffic flow or detection of special events. Location data sharing platforms such as Strava, GPSies, and OpenStreetMap (OSM) allow their users to share their location data with their community. In all of these cases, location measurements are considered collectively as sequences, each reflecting the movement of a person or a vehicle. Such sequences can be map-matched to paths in an underlying transportation network—in our case a *road network*—using appropriate methods [1]. We refer to such map-matched sequences as *trajectories* throughout the paper.

A common assumption is that most of the time, users travel on 'optimal' routes towards a (possibly intermediate) destination, where optimality is understood as the shortest path w.r.t. suitable scalar *traversal costs* of each road segment in the underlying road network. For instance, route planners and navigation systems often use travel times as traversal costs. However, in practice, drivers seldom travel on such 'optimal' routes due to complex traversal costs, e.g., time-dependent and uncertain travel times [2], a (possibly unknown) combination of several traversal costs [3], or due to changing intentions/destinations during a trip. We therefore investigate analysis techniques that do not rely on a fixed criterion but are capable of identifying a suitable combination of given criteria.

The high-level goal of this paper is to develop trajectory mining techniques to enable a better understanding of the semantics of trajectory data. Concretely, we focus on the tasks of trajectory segmentation and driving preference mining.

Trajectory Segmentation

A trajectory is often not just the manifestation of someone going from location S to location T following an optimal route w.r.t. some criteria, but rather determined by a sequence of activities/intentions. For instance, Figure 1 shows a trajectory from S to T with two intermediate stops labeled B. The driver starts at location S but rather than taking the fastest routes, decides to drive southwest and makes a stop. Then, the driver backtracks and takes the fastest route from S to T but decides to make a stop on the way. In this paper, we present a trajectory segmentation approach that can identify intermediate stops or other points of interest in a trajectory and divide it into subtrajectories accordingly.

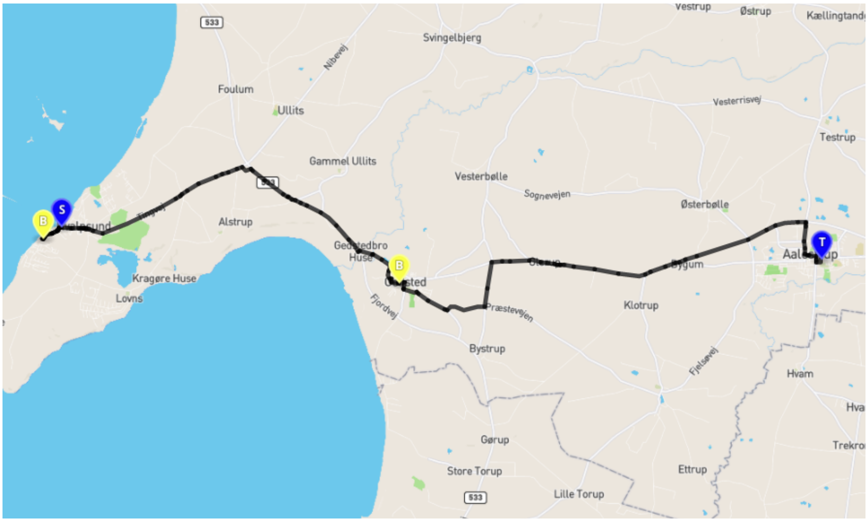


Fig. 1: An example of a trajectory going from S to T with two intermediate stops that are labeled B.

In contrast to previous work on trajectory segmentation [4–8], our approach solely relies on traversal costs and the structure of the road network. No additional information such as time stamps are required. Thus, compression techniques for efficient trajectory storage [9, 10] are applicable. However, despite not utilizing temporal information, our experiments show that our trajectory segmentation approach can recover such information through a structural analysis of the trajectory. In addition, our trajectory segmentation approach uses a driving preference model to divide trajectories into subtrajectories. To the best of our knowledge, this is the first trajectory segmentation approach to do so.

Driving Preference Mining

Given a trajectory, it is an interesting question which criteria the driver likely tried to optimize. We refer to this combination as a driving preference. We show how a known algorithm for driving preference mining [11] can be made sufficiently robust to recover preferences from vehicle trajectories. We compare the modified algorithm with two benchmark functions and address the question, how well the recovered preferences describe the trajectories.

Related Work

In this paper, we consider two main applications: trajectory segmentation and driving preference mining.

Previous work on trajectory segmentation can broadly be categorized as supervised [4, 5] and unsupervised [6–8]. Supervised approaches require predetermined criteria. In contrast, unsupervised approaches find the combination of available criteria that best explain the driving behavior in the input trajectory. Our proposed trajectory segmentation approach is unsupervised. Thus, the required prior assumptions on drivers' behavior are reduced to a minimum. In addition, our approach requires no spatio-temporal information about the trajectories, unlike existing trajectory segmentation approaches [4–8].

Driving preference mining—also referred to as driving preference learning —has been studied previously as well.

A popular approach of modeling driving preferences is to consider them as random variables. For instance, preference mining methods based on Gaussian Mixture Models are presented in [12] and in [13]. Balteanu et al. [14] present a probabilistic method that compares the input trajectory with pareto-optimal trajectory sets. Campigotta et al. [15] show how mining driving preferences can be accomplished using Bayesian learning strategies.

Our approach is non-probabilistic and considers the driving preference as the solution of an optimization problem. Delling et al. [3] present a similar technique, where the driving preference is chosen such that the overlap of the given and the computed trajectory is maximal. The main difference to our approach is that Delling et al. focus on geographical similarities of trajectories while we define similarity in terms of their traversal costs.

The work of Funke et al. [11] is the most closely related work both w.r.t. to trajectory segmentation and driving preference mining. They present a method which decides whether there exists a conic combination of the traversal costs such that a given trajectory is optimal for this weighting—or *preference*—of the traversal costs, and outputs the preference. Their method can recover the driving preferences of synthetic trajectories where such a conic combination is guaranteed to exist. However, this is not guaranteed for real world trajectories due to changes in driver preferences within the trajectory or inaccuracy in the traversal costs. We extend their method to obtain robust preferences in case a trajectory is not optimal for any preference. In addition, we use their method in our trajectory segmentation approach to determine the start and end of a subtrajectory.

1.1 Our Contribution

We develop techniques to 'explain' route choices made by drivers based on several natural criteria and evaluate them on a large data set of real-world trajectories. In particular, we propose a simple method for unsupervised trajectory segmentation that is able to approximate locations where drivers change their intentions/destinations along their trajectories. In contrast to previous work, our approach does not require spatio-temporal information about the trajectories, which considerably reduces the storage requirements.

Additionally, we present a simple, yet effective modification of a driving preference mining technique that allows to estimate drivers' preference that is more robust with respect to noise or sporadic 'suboptimal' routing decisions.

Both our approaches are built on the algorithm presented in [11] which deduce driving preferences from trajectories. As a result, driving preferences are integral to both our trajectory segmentation and driving preference mining techniques, demonstrating the close relationship between these two applications. To the best of our knowledge, we are the first to show this relationship.

2 Preliminaries

2.1 Data Set

Road Network Data

We use a directed graph representation of the Danish road network [16] $G = (V, E)$ that has been derived from data provided by the Danish Business Authority and the OSM project. In this graph representation, V is a set of nodes, each of which represents an intersection or the end of a road, and E is a set of edges, each of which represents a directed road segment. The graph representation of the Danish road network contains the most important roads and has a total of 583 816 intersections and 1 291 171 road segments. In addition, each road segment has attributes describing their length and type (e.g., motorway) and each intersection has attributes that indicate whether they are in a city area, a rural area, or a summer cottage area. The data is further augmented with a total of 163 044 speed limits combined from OSM data and speed limits provided by Aalborg Municipality and Copenhagen Municipality [17].

Trajectory Data

We use a set of 1 306 392 vehicle trajectories from Denmark collected between January 1st 2012 and December 31st 2014 [18]. The trajectories have been map-matched to the graph representation of the Danish road network s.t. each trajectory is a sequence of traversed road segments $T = (e_1, \dots, e_n)$ where $e_i \in E$ for $1 \leq i \leq n$. In addition, each segment is associated with a time stamp and a recorded driving speed whenever the GPS data is sufficiently accurate. In this data set, a trajectory ends after its GPS position has not changed more than 20 meters within three minutes. See [18] for more details.

Trajectory Stitching A vehicle trajectory in the trajectory data set ends when the vehicle has not moved more than 20 meters within three minutes. However, in practice, a driver may choose a trajectory with several intermediate stops, for instance when visiting multiple supermarkets to go grocery shopping. We are interested in examining such trajectories. We therefore stitch temporally consecutive trajectories from the same vehicle together if there is less than 30 minutes difference between the end of the current trajectory to the start of the next. Each stitch thus indicates the end of a 3 to 33 minutes break in movement. We call these stitches *break points* that mark a temporal gap in the trajectory.

In many cases temporally consecutive trajectories are not connected due to imprecision or lack of GPS data. In such cases, we compute the shortest route from the destination of the current trajectory to the start of the next. If the shortest route is shorter than 200 meters or consists of at most one road segment, we stitch the trajectories. We continue attempting to join the stitched to the next trajectory until the next trajectory does not meet the stitching criteria. See Section A for further details.

From the original 1 306 392 trajectories we obtain 260 190 combined trajectories. Of these trajectories, 190 199 trajectories are stitched and contain break points.

2.2 Routing Cost Types

From the data sets described in Section 2.1, we derive a number of criteria that are a measure of the expected cost of taking a route. In our experiments, we use the following four cost types: travel time, congestion, crowdedness, and number of intersections. We normalize the average value of each cost type to one.

Travel Time Each road segment is associated with a fixed value that represents the time it takes to traverse the road segment. To derive travel time, we combine historical traversal data from the trajectory data set with travel time estimates from a pre-trained machine learning model [16]. See Section B.1 for further details.

Congestion We derive the congestion level on a particular road segment based on how close to the speed limit people tend to drive. The closer to the speed limit, the less congestion. Many road segments do not have a speed limit in our speed limit data set. In such cases, we use a simple OSM routing heuristic, see Section B.2.

Crowdedness This criterion measures how 'crowded' the surroundings along a vehicle trajectory are. We derive a crowdedness value for each road segments from the number of nearby road segments and points of interest OSM nodes. Further details can be found in Section B.3.

Number of Intersections The number of intersections visited in a trajectory, excluding the source intersection.

2.3 Personalized Routing

The notion of shortest path requires some distance or cost measure to compare paths (i.e., routes) in a road network. To this end, we use the notion of personalized routing from [19] which takes into account both multiple traversal cost types and driving preferences.

Personalized Cost

The personalized cost of a road segment e combines a d-dimensional *cost vector* $c(e) = (c_1(e), \ldots, c_d(e))$ and a d-dimensional *preference vector* $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d)$ where $\alpha_i \geq 0$ and $\sum \alpha_i = 1$. Each cost $c_i(e)$ for $1 \leq i \leq d$ is a measure of the cost to traverse road segment e and each preference α_i represent the drivers preference w.r.t. to minimizing cost $c_i(e)$. For instance, $c_1(e)$ and $c_2(e)$ may be the travel time and the number of intersections of the road segment e. Assuming equal scale of the costs, a preference vector of $(0.7, 0.3)$ indicates that a driver values travel time more than the number of intersections.

The personalized cost of a route $\pi = (e_1, e_2, \ldots, e_k)$ in a road network is $p(\pi \mid \alpha) = \sum_{i=1}^{k} p(e_i \mid \alpha)$ where $p(e_i \mid \alpha)$ is the personalized cost of road segment e_i and α represent the driving preferences of a particular driver. Given driving preferences α, the personalized cost of a road segment e is $p(e \mid \alpha) = \sum_{i=1}^{d} \alpha_i c(e)$. We call a path π from s to t *personalized path* if $\forall \pi' \in \Pi : p(\pi \mid \alpha) \leq p(\pi' \mid \alpha)$, where Π is the set of all paths from s to t.

Deducing Routing Preferences

Given a trajectory T in the personalized route setting a natural question to ask is if it is a personalized path for some preference α. The trajectory $T = v_0 v_1 \ldots v_{k-1}$ is a personalized path if a solution exists to the following LP with variables $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_d)$ representing driving preferences [11].

$$\begin{aligned} \textsc{Minimize} \quad & 1 \\ \textsc{subject to} \quad & \forall \pi \in \Pi\colon p(T \mid \alpha) - p(\pi \mid \alpha) \leq 0 \\ & \sum_{i=1}^{d} \alpha_i = 1 \\ & \forall i \in \{1, \ldots, d\}\colon \alpha_i \geq 0 \end{aligned} \tag{E.1}$$

The LP contains a constraint for *each* possible route $\pi \in \Pi$ from v_0 to v_{k-1}. These constraints require the personalized costs of T for the solution, the preference vector α, to be lower than those of every other path π. Note that no objective function is used and T is therefore a personalized path if any preference vector α exists that satisfies all the constraints.

The Dijkstra Oracle The solution to the LP in Equation E.1 is a preference vector α for which no path from v_0 to v_{k-1} has lower personalized cost than the trajectory T. Writing down the complete LP for *all possible* paths from v_0 to v_{k-1} is infeasible. Fortunately, it suffices to add the constraints one by one via a so-called separation oracle [11].

In brief, the LP is first solved using only the last two constraints in Equation E.1. This results in some initial preference vector α. Then, α is verified to satisfy all the constraints in Equation E.1 by simply running a Dijkstra from v_0 to v_{k-1} with preference vector α. If so, a solution to the linear program in Equation E.1 has been found and no further processing is required. If not, a violating constraint is discovered and added to the LP.

While this method finds a preference method α for T, if any exists, real world trajectories are often not personalized paths.

2.4 Trajectory Segmentation

In this section, we discuss the definition of the trajectory segmentation problem and a general algorithmic framework for it.

Trajectory Segmentation Problem

The segmentation of a trajectory $T = v_0 v_1 \ldots v_{k-1}$ is a sequence of B trajectory segments $S_1 = v_0 v_1 \ldots v_{b_1}$, $S_2 = v_{b_1} v_{b_1+1} \ldots v_{b_2}$, $S_3 = v_{b_2} v_{b_2+1} \ldots v_{b_3}$, up to $S_B = v_{b_{B-1}} v_{b_{B-1}+1} \ldots v_{b_B}$. We refer to the common node of two consecutive trajectory segments, e.g., S_1 and S_2, as a *segmentation point*. For instance, v_{b_1} is a segmentation point because it is at the end of S_1 and the start of S_2. Buchin, et al. [4] define the segmentation problem as finding a (minimal) number of segments for a trajectory such that each segment fulfills a criterion. They provide a general algorithmic framework for arbitrary segmentation criteria.

Trajectory Segmentation Framework

In the framework of Buchin et al. [4], one has to provide a test procedure which verifies if a given segment meets the desired criterion. This test procedure is then used to repeatedly, greedily find the longest prefix that meets the criterion. The authors prove that this approach leads to an optimal, i.e., minimal, segmentation for *monotone* criteria in $O(T(k) \log k)$ time if the test procedure takes $T(m)$ time for a segment of length m. They define monotonicity for a criterion as follows. If any segment $S \subseteq T$ satisfies the criterion, then any segment $S' \subseteq S$ also satisfies the criterion. Even though Buchin et al. [4] focus on self-similarity criteria for the segments, this definition adapts well to the optimality criterion we introduce in Section 3.1.

3 Multi-Criteria Trajectory Segmentation

A driver may have several via-points on a trip. Sometimes, these via-points are linked to a point-of-interest such as gas station, but that is not necessarily the case. The trajectory segmentation approach we present in this section is designed to find all interesting via-points along a trajectory.

In brief, our approach assumes that drivers choose personalized paths between their via-points. Any deviation from their personalized path along a trajectory that goes from s to t indicates some interesting point p in the trajectory. The point of deviation p is marked as the end of the first trajectory segment and the beginning of the next. This process is repeated on the remaining subtrajectory going from p to t and so forth.

3.1 The Personalized Path Criterion

For trajectory segmentation in the framework of [4], described in Section 2.4, we propose a type of criteria that uses only the underlying graph and does not need any predetermined parameters. The *optimal path criterion* requires each trajectory segment S of a trajectory T to be an optimal path according to the traversal costs in the underlying graph. This criterion is monotone as defined in Section 2.4 because it requires S to be a "shortest path" and subpaths of shortest paths are also shortest paths. As a test procedure for a segment, we use a Dijkstra query.

The optimal path criterion can be generalized to the *personalized path criterion*. The personalized path criterion requires each trajectory segment to be a personalized path with respect to some driver preference α. This criterion is satisfied if there exists a solution to the LP in Equation E.1. Note, that the α for each trajectory segment can differ.

Fixing Edge Cases It is possible that there exists a minimal trajectory segment $S \subseteq T$ (consisting of a single road segment) which is not a personalized/optimal path. One road segment (u, v) might be more expensive in every traversal cost type than another path from u to v. This indicates that the used traversal cost types can not explain driver behavior for taking such an road segment. For the personalized path criterion, this can be remedied by including a cost type for which each road segment is a personalized path between its source and target intersections. In general, a unit cost type (every road segment e has $c_i(e) = 1$) has this property. This guarantees segmentability for arbitrary trajectories and makes the personalized path more robust. In our experiments, the number of intersections cost type fulfills this role.

This does, however, not fix the special case of self-loop edges, which in our data set typically represent road segments that allow traversals in parking lots. Such road segments can never be optimal because the optimal path from the source intersection to itself remains at the intersection. Self-loop edges can either be dealt with by deleting them from the trajectories, if they do not cover significant areas in the road networks, or by representing such road segments as two edges that each represent partial traversal of the self-looping road segment.

3.2 Experiments

We now investigate the capabilities of the trajectory segmentation method to identify via-points in trajectories on the basis of the trajectory data set described in Section 2.1. In particular, we use the stitched trajectory set to evaluate our trajectory segmentation approach, Personalized Path Trajectory Segmentation (PPTS), to the Optimal Path Trajectory Segmentation (OPTS) baseline which uses only a single cost type to check for the optimal path criterion. We consider the four variants OPTS-TT, OPTS-Con, OPTS-Int, and OPTS-Cro, that use the travel time, congestion level, number of intersections, and crowdedness, respectively, as the single cost type.

We compare PPTS's ability to segment trajectories to that of the baselines. Our comparison is both in terms of the number of trajectories that can be segmented and the ability of the trajectory segmentation algorithms to recover the break points in the stitched trajectory set. As mentioned in Section 2.1, these break points indicate a break of 3 to 33 minutes and are therefore likely to indicate a via-point within the trajectory. We discard the self-loop edges within each trajectory to increase segmentability, as described in Section 3.1. Typically, these self-loop edges represent road segments that allow driving around parking lots.

All algorithms used in our experiments are implemented in the Rust programming language[1]. We make the implementation of our method, the used graph and some example trajectories publicly available[2]. We use contraction hierarchies (CH) [20] to speed up the Dijkstra queries by orders of magnitude.

Evaluation Functions

We use several evaluation functions to evaluate our trajectory segmentation method and for comparison with the baselines.

Segmentability Score The segmentability score, or simply S-score, measures the proportion of trajectories that are segmentable by a trajectory segmentation algorithm. Ideally, the S-score is 100% indicating that all trajectories in the data set could be segmented by the used trajectory segmentation algorithm.

Break Recovery Rate The Break Recovery Rate (BRR) is a measure of how good a trajectory segmentation algorithm is at placing segmentation points s.t. they coincide with known break points in the stitched trajectories. Let BP denote the set of known break points in a trajectory T and let SP denote the set of segmentation points output by a trajectory segmentation algorithm that has been given trajectory T as input. Then, the BRR of trajectory T is

$$\text{BRR}(T) = \frac{|RBP|}{|BP|}$$

where $RBP = BP \cap SP$ is the set of recovered break points.

Segmentation Rate A trajectory segmentation algorithm can achieve a high BRR by simply segmenting a trajectory into trajectory segments consisting of one road segment each. Although such a segmentation is guaranteed to recover all break points, it is also very likely to contain a lot of noise in the form of many false positives or false break points. To measure such noise, we use the Segmentation Rate (SR) which measures the number of segmentation points per break point:

$$\text{SR}(T) = \frac{|SP|}{|BP|}$$

Ideally, the SR should be 1 for a trajectory segmentation that recovers all break points, i.e., has a BRR of 100%.

[1] `https://www.rust-lang.org/`

[2] `https://github.com/Lesstat/ppts`

Table 1: Mean algorithm performance on all stitched trajectories (ALL) and the **60, 249** commonly segmentable trajectories (CS) that can be segmented by all algorithms.

	ALL		*CS*		
Algorithm	*BRR*	*S-score*	*BRR*	*SR*	*SQ-score*
PPTS	57.98%	**100.0%**	56.29%	**2.39**	**0.235**
OPTS-TT	34.62%	58.16%	**58.35%**	2.83	0.206
OPTS-Con	29.32%	49.61%	57.49%	3.99	0.144
OPTS-Int	**59.19%**	**100.0%**	57.61%	4.37	0.132
OPTS-Cro	35.86%	61.11%	58.10%	5.62	0.103

Segmentation Quality Score The segmentation quality score, or simply SQ-score, is a summary score to measure the overall quality of a trajectory segmentation. It combines the BRR and SR as follows:

$$\mathrm{SQ}(T) = \frac{\mathrm{BRR}(T)}{\mathrm{SR}(T)}$$

Note, that the unit of the SQ-score is recovered break points per segmentation point and should ideally be 1.

Results

The results of our experiments are shown in Table 1.

Segmentability As shown in Table 1, PPTS and OPTS-Int are both capable of segmenting all trajectories and achieve an S-score of 100%. This result is not too surprising, since both algorithms use a unit cost type—the number of intersections—which guarantees that any trajectory can be segmented by these approaches, as discussed in Section 3.1. The remaining algorithms cannot segment a large portion of the trajectories (more than half in the case of OPTS-Con) and therefore achieve comparatively low S-scores. Thus, the inclusion of additional cost types can increase segmentability.

Segmentation Quality As shown in Table 1, PPTS and OPTS-Int achieve similar BRRs that are substantially higher than the remaining OPTS variants. However, for a fair comparison that ignores the ability of the algorithms to segment trajectories, we have computed BRRs, SRs, and SQ-scores on the subset of trajectories that are commonly segmentable, i.e., the trajectories that can be segmented by all algorithms. On this subset, the BRRs of all algorithms are comparable. This suggests that the superior BRR when considering all trajectories for PPTS and OPTS-Int can largely be attributed to their greater capability for segmenting trajectories.

Although the BRRs are quite similar on the commonly segmentable trajectories, the SRs are quite different, as shown in Table 1 In particular, the OPTS-Con, OPTS-Int, and OPTS-Cro algorithms have, respectively, 67%, 83%, and 135% more segmentation points per break point than PPTS. The SR of OPTS-TT algorithm is just 18% higher than that of PPTS.

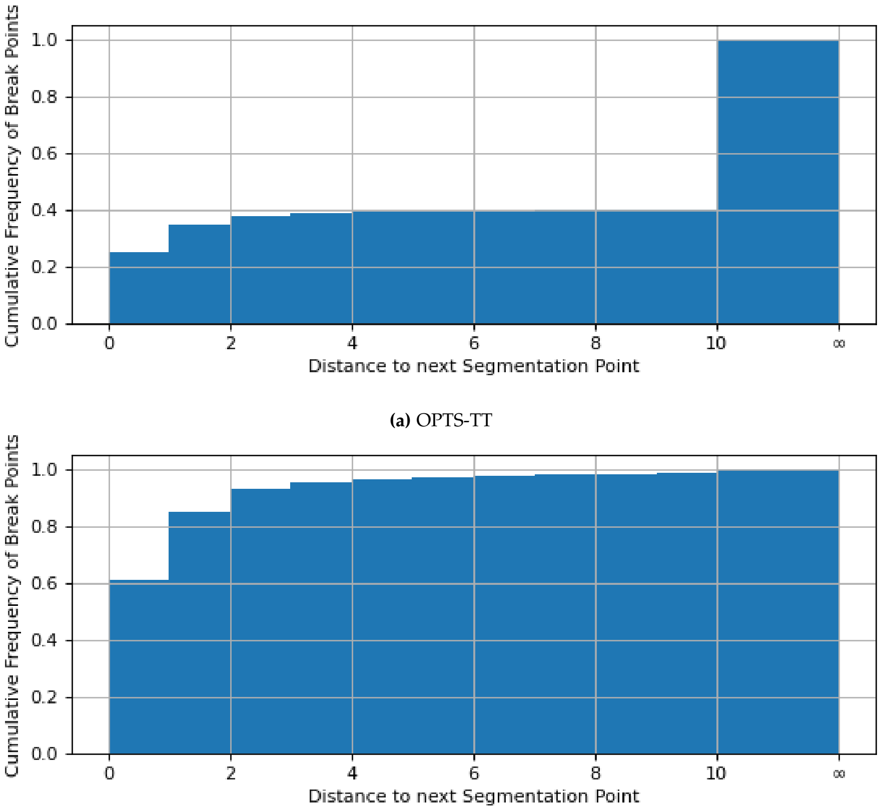
The higher SRs of the single-cost-type baselines compared to PPTS suggest that the inclusion of driving preferences and multiple criteria reduces the amount of false positives. The PPTS achieves the lowest BRR on the commonly segmentable trajectories, but, as shown in Table 1, PPTS achieves the best overall trajectory segmentation quality with an SQ-score of 0.235, since it introduces the fewest false break points and thus have the lowest SR. Conversely, OPTS-TT achieves the highest BRR score on the same data subset, but introduces more false break points than PPTS. As a result, OPTS-TT achieves only the second-highest segmentation quality with an SQ-score of 0.206. Still, our results support the wide-spread use of the travel time cost type in many routing services, but also show that taking additional cost types and driving preferences into account can lead to better trajectory segmentation.

For the sake of brevity, we present only the comparison between PPTS and the best-performing baseline, OPTS-TT, in the remainder of this section.

Segmentation Point Accuracy We have thus far only considered exact break point recovery, but a segmentation may still be useful if it indicates that a break point is near. Figure 2 shows the percentage of break points which is within a certain (hop) distance to the next segmentation point for OPTS-TT and PPTS. PPTS places segmentation points considerably more accurate than OPTS-TT. PPTS places more than 60% of the break points within one road segment of the nearest segmentation point and over 80% are within two road segments of the next segmentation point. OPTS-TT achieves less than half of PPTS's performance. However, the performance disparity illustrated in Figure 2 is largely due to better segmentability of trajectories when using PPTS. If the analysis is restricted to break points with distance $d < \infty$ to the nearest segmentation point, i.e., trajectories that are segmentable by OPTS-TT, their distributions are comparable.

Qualitative Segmentation Assessment A good trajectory segmentation marks break points (or other interesting points) along a trajectory with a segmentation point. However, a good trajectory segmentation should also avoid too many false positives.

The PPTS and OPTS-TT, respectively, have an SR of 2.39 and 2.83 segmentation points per break point, respectively. However, although these numbers suggest that there are more false break points when using OPTS-TT, our data only contains positive examples of interesting behavior within the trajectory,



(a) OPTS-TT

(b) PPTS

Fig. 2: Distribution of distance between a break point and the next segmentation point for (a) OPTS-TT and (b) PPTS. Break points in trajectories without any segmentation point are assigned distance **∞**.

i.e., the break points in the stitched trajectories. As result, we cannot quantitatively determine whether the segmentation points that do not match a break point are indeed false positives or mark interesting, but unknown, behavior during the trajectory. We therefore qualitatively assess the validity of the segmentation of a few trajectories.

Figure 3 shows a break point (marked with 'B' in yellow) in a segmented trajectory. The segmentation by OPTS-TT shown in Figure 3a places two segmentation points (marked with 'S' in black) around the break point. These segmentation points fail to recover the break point but are both within a distance of two road segments of the break point. Thus, the OPTS-TT segmentation appear to detect the presence of the break point, but fails to place the segmentation points exactly. The PPTS segmentation shown in Figure 3b, is a better segmentation and recovers the break point exactly (indicated by the black marker labeled 'B').

Figure 4 shows another part of the trajectory shown in Figure 3. Here, OPTS-TT places a segmentation point without comparable segmentation points in the PPTS segmentation. This additional segmentation point has no apparent meaning, and, upon detailed inspection, appears to occur due to inaccuracies in the estimated travel time in the area. This suggests that PPTS may be more robust than OPTS-TT to noise in the traversal cost data.

For the purposes of quantitative evaluation, our method attempts to recover breaks of 3 to 33 minutes from trajectories. However, our trajectory segmentation approach can discover interesting behavior beyond these known breaks. For instance, Figure 5 shows a segmentation point marking a detour to a gas station. This segmentation point is placed by both OPTS-TT and PPTS.

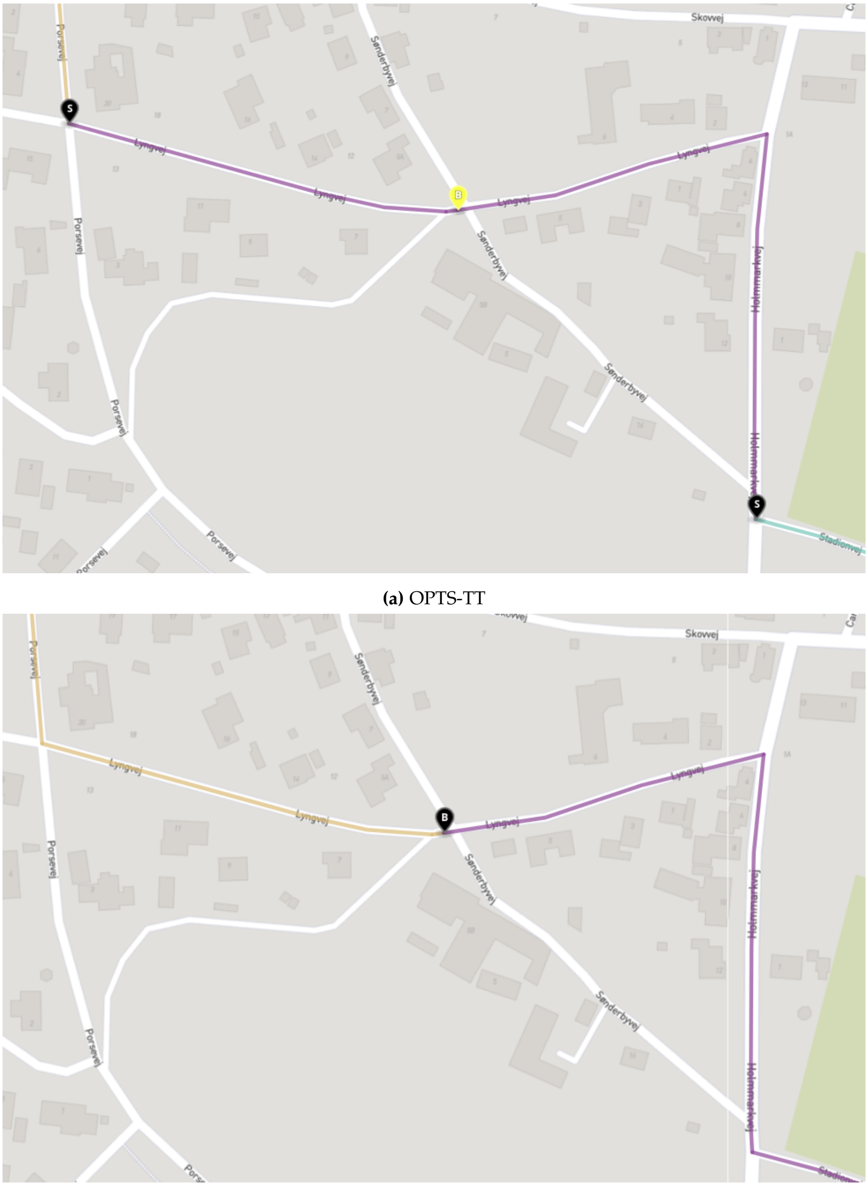
Processing Time

While using personalized routing does improve break recovery, it comes with an increase in processing time of trajectory segmentation. The increase in processing time for the personalized path variant is mostly driven by the CH-Dijkstra queries being slower.

The trajectory segmentation process is trivially parallelizable, since each trajectory can be processed independently, making segmentation of even billions of trajectories feasible. In our experiments, we parallelized the trajectory segmentation process across 64 cores, each with a clock speed of 2.3 GHz. The time to process the 190, 199 stitched trajectories for single-criteria and multi-criteria trajectory segmentation is, respectively, 1 and 5 hours in total, and 19 and 95 milliseconds per trajectory on average. The total processing time took about half an hour in wall-clock time.

3.3 Discussion

Overall, PPTS achieves the highest trajectory segmentation quality in our experiments, followed by OPTS-TT. The results suggest that PPTS has two primary advantages over the baselines in our experiments. The use of multiple cost types and driving preferences makes PPTS capable of (1) segmenting more trajectories and (2) explaining driving behavior better, resulting in fewer false break points being placed. Our qualitative assessment of the OPTS-TT and PPTS segmentations support the conclusion that the segmentation points placed by PPTS are less likely to be false positives. In addition, PPTS discovered a detour to a gas station that is not indicated by a break point in our trajectory data set.



(a) OPTS-TT

(b) PPTS

Fig. 3: A break point in a trajectory and the segmentation points for (a) OPTS-TT and (b) PPTS. Yellow markers labeled 'B' indicate a break point and black markers labeled 'S' a segmentation point. A black marker labeled B indicates a break point that is recovered by a segmentation point.

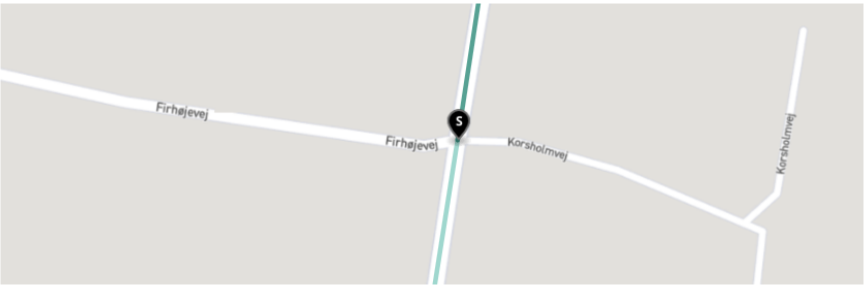


Fig. 4: A segmentation point with no obvious event occurring.

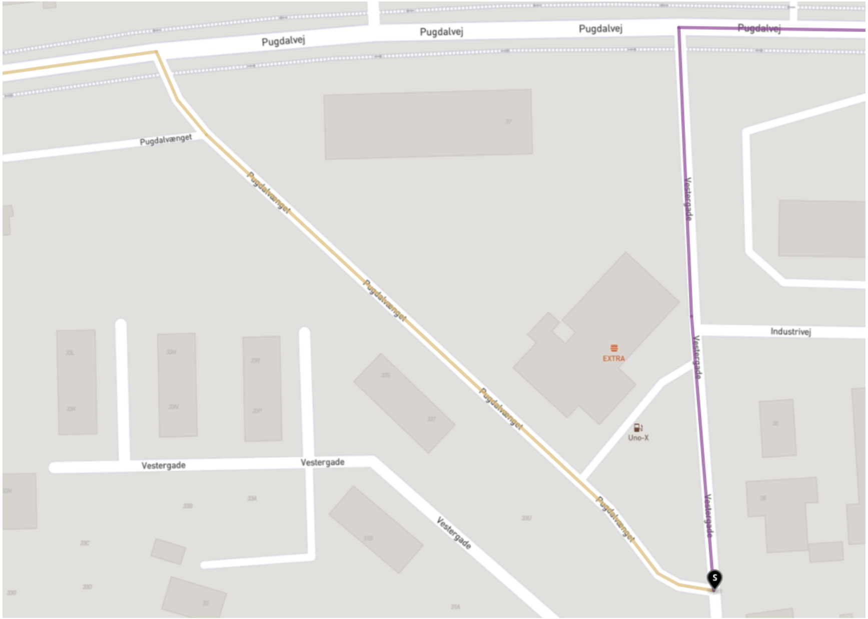


Fig. 5: A segmentation point recovers a detour to a gas station that is not marked as a break in our data set.

Even in the case where a break point is not recovered, PPTS is likely to place a segmentation point near the break point. PPTS places a segmentation point within a distance of 3 road segments for 95% of break points, but OPTS-TT for less than 40% of the break points. Although, this difference is largely explained by PPTS being capable of segmenting more trajectories, it suggests that PPTS's segmentation points are likely to indicate some interesting part of a trajectory. Although the increase in performance of PPTS over OPTS-TT comes at a factor 5 increase in processing time, it is still capable of segmenting a trajectory in a fraction of a second on average.

Stitching Parameters Changing the parameters for the stitching process has only very little effect on our results and do not affect our conclusions. The results are virtually invariant to changes to the temporal stitching threshold. However, they are sensitive to changes to the stitch length threshold, although the effect is minor. The longer the stitches are allowed to be, the worse break recovery performance for both OPTS-TT and PPTS. This is likely because more noise is introduced when longer stitches are allowed.

4 Robust Driving Preference Mining

The trajectory segmentation approach presented in Section 3 implicitly recovers driving preferences α for each trajectory segment when determining the personalized path segments by searching for a solution to Equation E.1. Although it is possible to collect the preference vectors for each trajectory segment, one is typically more interested in a *single* preference vector to describe a driver's general behavior for use in, e.g., personalized route planning [12, 14, 19]. Fortunately, this is possible with only a minor modification of the linear program in Equation E.1:

$$\begin{array}{ll}
\textsc{Minimize} & \delta \\
\textsc{subject to} & \forall \pi \in \Pi \colon p(T \mid \alpha) - p(\pi \mid a) \leq \delta \\
 & \displaystyle\sum_{i=1}^{d} \alpha_i = 1 \\
 & \forall i \in \{1, \dots, d\} \colon \alpha_i \geq 0 \\
 & \delta \geq 0
\end{array} \tag{E.2}$$

As with Equation E.1, this linear program may also be solved in polynomial time using the LP path oracle described in Section 2.3.

The following modifications have been made to Equation E.2. First, the solution to the linear program is a preference vector α for the whole trajectory T rather than a trajectory segment. Second, by introducing δ to the first constraint, T is not required to be a personalized path w.r.t. to the α. Third, Equation E.1 minimizes δ s.t. T is as close to being shortest-path optimal as possible w.r.t. to the preference vector α that is the solution to the linear program.

The effect of the modifications made in Equation E.2 is that the linear program always has a feasible solution and therefore our approach always outputs some preference vector α. If $\delta = 0$, then the recovered α fully explains the driver behavior in the trajectory and is identical to the solution of Equation E.1. Otherwise, if $\delta > 0$, then the recovered α does not fully explain the driver behavior but explains it as much as possible given the available traversal costs.

4.1 Experiments

We now evaluate our driving preference mining approach on a personalized routing task using the data set described in Section 2.1. Specifically, we evaluate our approach for each trajectory $T = v_0 v_1 \ldots v_{k-1}$ in a trajectory set as follows. First, we solve Equation E.2 for T. Then, we use the resulting preference vector α to compute a personalized route (or personalized path) π from v_0 to v_{k-1}. Ideally, the preference vector α combined with the source v_0 and target v_{k-1} is sufficient to reconstruct or recover the route driven in trajectory T. We therefore refer to π as the *recovered route* of trajectory T.

W.r.t. the task of personalized routing, we are interested in measuring two qualities about our approach. First, how well do the recovered preference vectors model driving behavior. Second, how well do the preference vectors match the preferences of the drivers.

Evaluation Functions

To measure our approach's ability to model driving behavior, we use the Relative Recovered Route Overlap (RRRO):

$$\mathrm{RRRO}(T, \pi) = \frac{|T \cap \pi|}{|T|}$$

Let α be a preference vector recovered from trajectory T. Here, π is the recovered route of trajectory T. If the preference vector α used to construct π fully captures the driving preferences exhibited in T, then the route π recovered using α should be identical to T, resulting in a relative recovered route overlap of 1.

To measure whether the preference vectors found using our approach match (actual) driver preferences, we use the Relative Cost Recovery Score (RCRS). The RCRS reflects the view that two routes are equivalent if their

Table 2: Mean RRRO and RCRS of RDP, TTP, and BRP of different algorithms for personalised routing on the unstitched trajectory set.

	RDP	*TTP*	*BRP*
RRRO	**0.74**	0.70	0.66
RCRS	**0.87**	0.85	0.81

personalized costs are identical:

$$\mathrm{RCRS}(T, \pi) = \frac{p(\pi \mid \alpha)}{p(T \mid \alpha)}$$

Note, that $p(T \mid \alpha) \geq p(\pi \mid \alpha)$ since π is the shortest path w.r.t. the personalized costs of the road segments for the given preference vector α. Thus, the RCRS is always between 0 and 1. An RCRS value of, e.g., 0.8, indicates that the preference vector α accounts for 80% of the personalized cost of trajectory T. If the preference vector α fully captures the driver's preferences, then the RCRS is 1.

Baselines

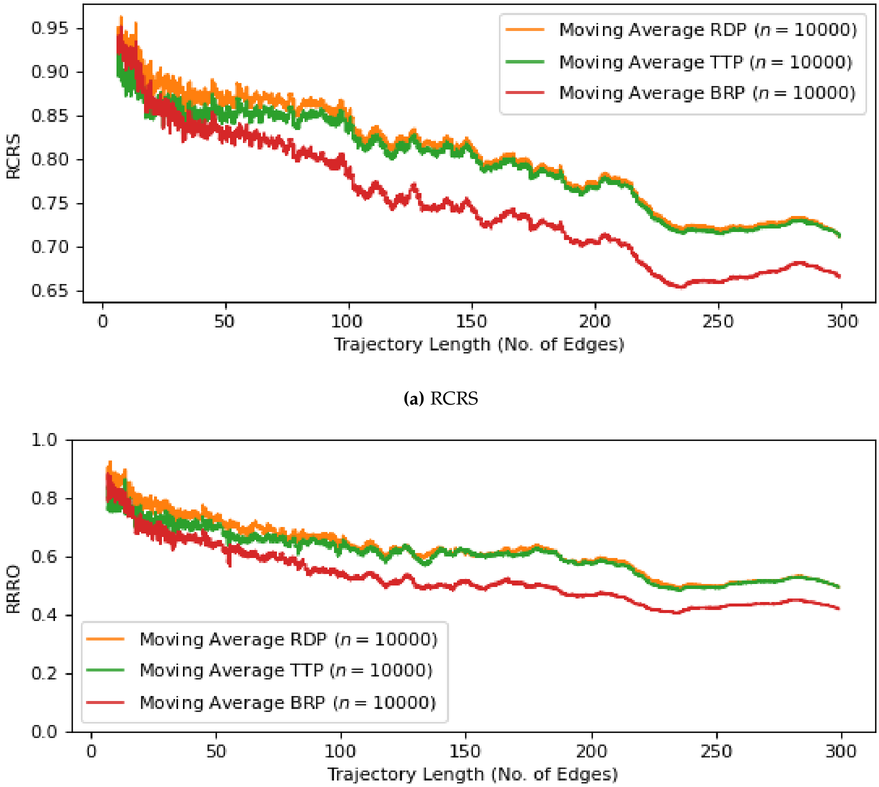
We refer to our approach as Recovered Driving Preference (RDP) and compare its performance with two baselines. The first baseline, Travel Time Preference (TTP), always returns a preference vector that has weight one for travel time. The second baseline, Best Random Preference (BRP), generates five random preference vectors for a trajectory, evaluates them and returns the preference with the best result. The BRP baseline is run independently for the two evaluation functions used in our experiments.

Results

We run our experiment on both the unstitched and stitched trajectory sets. The results on both trajectory sets are similar. For brevity, we report only the results on the unstitched trajectory set.

We summarize the results on the unstitched trajectories in Table 2. As shown in the table, RDP achieves both the highest mean RRRO and mean RCRS that are, respectively, 5.7% and 2.4% better than the best performing baseline TTP. We expect that these figures will be even higher if more than four traversal cost types are used.

The RDP shows superior performance compared to both baseline algorithms, but the performance of both RDP and the baselines deteriorate as trajectory length increases as shown in Figure 6. This is not particularly surprising since longer trajectories are more likely to contain more via-points,



(a) RCRS

(b) RRRO

Fig. 6: The (a) RCRS and (b) RRRO scores of RDP, the TTP and the BRP with unstitched trajectories.

and none of the approaches in this experiment take such information into account. In addition, TTP approaches the performance of RDP as trajectory length increases. This suggests that people are likely to prioritize travel time more on long trips which matches both our expectations and anecdotal experiences.

Robustness

We call our algorithm robust because, in contrast to the original algorithm given in [11], it does not require the trajectory to be a personalized path. Therefore, we are able to recover a driving preference for each of the 1, 306, 392 trajectories. On the contrary, the baseline algorithm is only able to process 590, 251 trajectories, which is less than half. Hence, our modification is indeed a considerable robustness improvement.

Processing Time

We computed the results with a single core with a clock speed of 2.3 GHz. The average processing time is 1.04 milliseconds per trajectory and 0.02 milliseconds per road segment, but is proportional to the number of road segments in the trajectory.

4.2 Discussion

Our experiments show that the RDP preference vectors explain driver behavior better than the TTP and BRP baselines. Although the average results of TTP are very similar to those of RDP, they are never better. This is a strong indication that our approach indeed finds the best preferences to describe the drivers' behavior. In addition, the fast processing time (and trivial parallelizability) of RDP makes it scalable to even very large trajectory data sets.

Notably, the performance gap between RDP and TTP nearly disappears for long trajectories. This matches our expectation that travel time is the most important criterion for such trajectories.

5 Conclusion

In this paper, we have presented two techniques for large scale trajectory segmentation and driver preference mining. We have shown experimentally that our proposed trajectory segmentation approach is a useful tool for understanding the semantics of a trajectory, e.g., the driver's intentions or changing destinations. In addition, our experiments showed that our proposed driver preference mining technique can indeed discover driver preferences for real trajectories and is sufficiently robust to process such data. Our techniques can be implemented efficiently in practice and are trivially parallelizable. Thus, they scale to very large trajectory sets consisting of millions or even billions of trajectories.

Interestingly, our approaches for trajectory segmentation and driving preference mining rely on the same model of driving preferences, showing that these two tasks are closely linked. To the best of our knowledge, we are the first to show this link.

Future Directions There are many interesting directions for both our trajectory segmentation and driver preference mining approaches.

Our trajectory segmentation approach relies on linear combinations of costs. However, relationships between costs may be more complex and present an interesting opportunity for future work. In addition, driver preferences can be highly context-dependent and depend on, e.g., the time of day [12]. Extending our approach to utilize such contextual information is an important future direction.

In our driver preference mining approach, the linear program may have multiple solutions corresponding to a large set of preferences or a preference space. This makes it difficult to compare recovered preferences among trajectories or trajectory segments of the same trajectory. In particular, solving the linear program for two trajectories generated by two drivers with the same preferences may yield two different solutions, even if they also follow the same route. An important future direction is therefore to extend our driving preference mining technique to output identical (or at least similar) preferences in such situations. This will enable analysis of driver behavior through, e.g., driver preference clustering.

Finally, our driving preference mining approach assumes that the intent of the driver is to go straight from the start of the trajectory to the end of the trajectory. Thus, it ignores that such trajectories may have via-points. However, as demonstrated by our experiments, our trajectory segmentation approach can discover such via-points in trajectories. In future work, it would be interesting to explore synergies between our trajectory segmentation and our driving preference mining approach.

Acknowledgments

This work was supported in part by the DiCyPS project, by grants from the Obel Family Foundation and VILLUM FONDEN, and in part by the Deutsche Forschungsgemeinschaft (DFG) within the priority program 1894: Volunteered Geographic Information.

References

[1] Y. Zheng, “Trajectory data mining: An overview,” *ACM Trans. Intell. Syst. Technol.*, vol. 6, no. 3, pp. 29:1–29:41, May 2015.

[2] S. A. Pedersen, B. Yang, and C. S. Jensen, “Fast stochastic routing under time-varying uncertainty,” *The VLDB Journal*, vol. 29, no. 4, pp. 819–839, 2020.

[3] D. Delling, A. V. Goldberg, M. Goldszmidt, J. Krumm, K. Talwar, and R. F. Werneck, “Navigation made personal: Inferring driving preferences from gps traces,” in *Proc. of SIGSPATIAL’15*. New York, NY, USA: Association for Computing Machinery, 2015.

[4] M. Buchin, A. Driemel, M. Van Kreveld, and V. Sacristán, "Segmenting trajectories: A framework and algorithms using spatiotemporal criteria," *Journal of Spatial Information Science*, vol. 2011, no. 3, pp. 33–63, 2011.

[5] S. P. Alewijnse, K. Buchin, M. Buchin, A. Kölzsch, H. Kruckenberg, and M. A. Westenberg, "A framework for trajectory segmentation by stable criteria," in *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2014, pp. 351–360.

[6] A. Soares Júnior, B. N. Moreno, V. C. Times, S. Matwin, and L. d. A. F. Cabral, "GRASP-UTS: an algorithm for unsupervised trajectory segmentation," *International Journal of Geographical Information Science*, vol. 29, no. 1, pp. 46–68, 2015.

[7] A. S. Junior, V. C. Times, C. Renso, S. Matwin, and L. A. Cabral, "A semi-supervised approach for the semantic segmentation of trajectories," in *2018 19th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2018, pp. 145–154.

[8] M. Etemad, A. S. Júnior, A. Hoseyni, J. Rose, and S. Matwin, "A trajectory segmentation algorithm based on interpolation-based change detection strategies." in *EDBT/ICDT Workshops*, 2019.

[9] B. Krogh, C. S. Jensen, and K. Torp, "Efficient in-memory indexing of network-constrained trajectories," in *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2016, pp. 1–10.

[10] R. Song, W. Sun, B. Zheng, and Y. Zheng, "PRESS: A Novel Framework of Trajectory Compression in Road Networks," in *Proceedings of the VLDB Endowment*, vol. 7, may 2014, pp. 661–672.

[11] S. Funke, S. Laue, and S. Storandt, "Deducing individual driving preferences for user-aware navigation," in *Proc. of SIGSPATIAL'16*, 2016.

[12] B. Yang, C. Guo, Y. Ma, and C. S. Jensen, "Toward personalized, context-aware routing," *The VLDB Journal*, vol. 24, no. 2, pp. 297–318, Apr. 2015. [Online]. Available: http://dx.doi.org/10.1007/s00778-015-0378-1

[13] J. Dai, B. Yang, C. Guo, and Z. Ding, "Personalized route recommendation using big trajectory data," in *2015 IEEE 31st international conference on data engineering*. IEEE, 2015, pp. 543–554.

[14] A. Balteanu, G. Jossé, and M. Schubert, "Mining driving preferences in multi-cost networks," in *SSTD*, ser. Lecture Notes in Computer Science, vol. 8098. Springer, 2013, pp. 74–91.

[15] P. Campigotto, C. Rudloff, M. Leodolter, and D. Bauer, "Personalized and situation-aware multimodal route recommendations: The favour algorithm," *Trans. Intell. Transport. Sys.*, vol. 18, no. 1, pp. 92–102, Jan. 2017.

[16] T. S. Jepsen, C. S. Jensen, and T. D. Nielsen, "Relational Fusion Networks: Graph Convolutional Networks for Road Networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–12, 2020, in online early access.

[17] T. S. Jepsen, C. S. Jensen, T. D. Nielsen, and K. Torp, "On network embedding for machine learning on road networks: A case study on the danish road network," in *Proc. of Big Data 2018*, 2018, pp. 3422–3431.

[18] O. Andersen, B. B. Krogh, and K. Torp, "An Open-source Based ITS Platform," in *Proc. of MDM*, vol. 2, 2013, pp. 27–32.

[19] S. Funke and S. Storandt, "Personalized route planning in road networks," in *Proc. of SIGSPATIAL'15*, 2015, pp. 1–10.

[20] S. Funke, S. Laue, and S. Storandt, "Personal Routes with High-Dimensional Costs and Dynamic Approximation Guarantees," in *16th Int. Symp. on Experimental Algorithms (SEA 2017)*, ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 75, Dagstuhl, Germany, 2017, pp. 18:1–18:13.

[21] M. Fruensgaard and T. S. Jepsen, "Improving cost estimation models with estimation updates and road2vec: a feature learning framework for road networks," Master's thesis, Aalborg University, 2017.

Appendices

A Trajectory Stitching

Our trajectory dataset $D = \{\mathbb{T}_1, \ldots, \mathbb{T}_n\}$ consists of sets of trajectory sequences of the form $\mathbb{T}_i = (T_1, \ldots, T_m)$. Each trajectory sequence T_i contain trips specific to driver d_i and are in temporal order s.t. trajectory $T_j \in \mathbb{T}_i$ started before trajectory $T_{j+1} \in \mathbb{T}_i$. We define a stitched trajectory data set based on the data set D as $D_{stitched} = \bigcup_{i=1}^{n} \textsc{StitchTrajectories}(\mathbb{T}_i)$.

Algorithm 7 Trajectory Stitching

```
function $\textsc{StitchTrajectories}(\mathbb{T} = (T_1, \ldots, T_n))$
    $\mathbb{T}_{stitched} \leftarrow \emptyset$
    $T_{current} \leftarrow T_1$
    $e_{current} \leftarrow \textsc{GetEndTime}(T_{current})$
    for $i = 2$ to $n$ do
        $s_i \leftarrow \textsc{GetStartTime}(T_i)$
        if $s_i - e_{current} \leq 30$ minutes and
                $T_i$ and $T_{current}$ are pseudo-connected then
            $T_{current} \leftarrow \textsc{Stitch}(T_{current}, T_i)$
        else
            $\mathbb{T}_{stitched} \leftarrow \mathbb{T}_{stitched} \cup \{T_{current}\}$
            $T_{current} \leftarrow T_i$
        $e_{current} \leftarrow \textsc{GetEndTime}(T_{current})$
    $\mathbb{T}_{stitched} \leftarrow \mathbb{T}_{stitched} \cup \{T_{current}\}$
    return $\mathbb{T}_{stitched}$
```

The StitchTrajectories function, defined in Algorithm 7, takes as input the trajectories of a driver in temporal order. We use $T_{current}$ to keep track of the current trajectory considered for stitching. Initially, $T_{current}$ is set to T_1. We use $e_{current}$ to keep track of the end time of the current trajectory $T_{current}$, i.e., its last recorded GPS point. In a loop, we scan the input trajectories $\mathbb{T}$ sequentially for stitching candidates, starting from trajectory T_2. We first store the time of the first GPS point associated with trajectory T_i in s_i.

We then check whether T_i is both temporally and spatially near enough to stitch with $T_{current}$. The two trajectories $T_{current}$ and T_i are temporally near enough to stitch if there is at most a 30 minute difference between $e_{current}$ and s_i. Two trajectories $T_1 = (e_1, \ldots, e_i)$ and $T_2 = (e_{i+1}, \ldots, e_{i+j})$, where $e_i = (u, v)$ and $e_{i+1} = (w, x)$, are pseudo-connected if the shortest route between v and w consists of at most one road segment or is less than 200 meters in length. If both stitching conditions are met, T_i is stitched to $T_{current}$ by invoking the Stitch function.

For pseudo-connected trajectories T_1 and T_2, Stitch is defined as $\textsc{Stitch}(T_1, T_2) = (e_1, \ldots, e_i, e'_1, \ldots, e'_k, \ldots, e_{i+j})$ where $(e'_1, \ldots, e'_k)$ is the shortest route connecting v and w which we refer to as a *stitch*. Then, the stitched trajectory is assigned to $T_{current}$.

If the stitching conditions are not met, we cannot stitch more trajectories to $T_{current}$. We then add the current trajectory to $\mathbb{T}_{stitched}$ and let T_i be the new current trajectory. Note that after the first iteration $T_{current}$ may be a stitched trajectory. For a stitched trajectory $T' = \textsc{Stitch}(T_1, T_2)$, we define $\textsc{GetEndTime}(T') = \textsc{GetEndTime}(T_2)$. After scanning through all of the input trajectories, we add the last trajectory to $\mathbb{T}_{stitched}$ and finally return the stitched trajectories.

B Routing Cost Type Details

In this section, we describe how the travel time, congestion, and crowdedness routing costs are derived in further details.

B.1 Travel Time

The vehicle trajectories in our trajectory set have the tendency to be concentrated on a few popular segments, as such, many road segments have few or no traversals in the trajectory set. We therefore require a means of estimating travel times for such road segments. To this end, we use a pre-trained machine learning model to provide travel time estimates. However, for road segments with an abundance of traversal data the model's estimates may be inaccurate. Inspired by previous work [21], we therefore combine travel time estimates with travel times of historical traversals s.t. when the driving speed estimate of a road segment becomes increasingly less influential the more historical traversals the road segment is associated with.

We compute the travel time t_e for a road segment e as $t_e = \frac{k\hat{t}_e + n\bar{t}_e}{k+n}$ where $\hat{t}_e$ is the estimate of the mean travel time, $\bar{t}_e$ is the mean travel time of the historical traversals, n is the number of historical traversals of segment e in the trajectory dataset, and k represents the confidence in $\hat{t}_e$. We use $k = 10$ in our experiments.

We use a pre-trained Relational Fusion Network (RFN) [16] to provide travel time estimates $\hat{t}_e$ for each road segment $e \in E$. Specifically, we use the best performing RFN from [16] which has been trained on the Danish Municipality of Aalborg using trajectories within the municipality that occurred between January 1st 2012 and June 30th 2013. Despite having been trained only on a subset of the network, the model generalizes well to unseen areas of the road network [16]. However, in a few cases the network would give very low values. We therefore modify the output s.t. the estimated driving speed on any road segment cannot be below 5 kmh.

B.2 Congestion

We assign a congestion level to road segment e depending on the speed limit s_e on the segment in km/h, the length of l_e of the segment in km, and the travel time t_e in hours. Let $\tau_e = l_e/s_e$ denote the travel time on road segment e if a vehicle is driving at exactly the speed limit. Formally, we assign road segment e the congestion level $c_e = \max\{1 - \frac{t_e}{\tau_e}, 0\}$ s.t. a value of 0 indicates that it is possible to drive at (or above) the speed limit and a value of 1 indicates that the road segment is not traversable.

The value of τ_e relies on the speed limit of road segment e. We use a speed limit data set that combines OSM speed limits with speed limits provided by Aalborg Municipality and Copenhagen Municipality [17]. This data set contains 163 044 speed limits, thus leaving many road segments without a known speed limit. In such cases, we use an OSM routing heuristic[3] which in Denmark assigns a speed limit of 130 km/h to motorways, a speed limit of 50 km/h in cities, and a speed limit of 80 km/h on other types of segments. For our data, we count a road segment as in a city if either the source or destination intersection is in a city according to its attributes.

B.3 Crowdedness

This routing cost type describes how 'crowded' the landscape around a road segment is. It is derived from the number of OSM nodes in the vicinity of the road segment. We use all OSM nodes in Denmark from a 2019 data set regardless wether they represent a road, a building or some other point of interest. To calculate it, we first overlay our graph with a grid and count the OSM nodes within each cell. For each road segment, we locate the OSM nodes that are part of its geometry in the grid. The cost per road segment is then the sum of the cell counts of its (geometry) nodes. We use a grid of 2000 by 2000 resulting in a cell size of roughly 209m x 177m.

[3] See `https://wiki.openstreetmap.org/wiki/OSM_tags_for_routing/Maxspeed`.

ISSN (online): 2446-1628
ISBN (online): 978-87-7210-977-0

AALBORG UNIVERSITY PRESS