Aalborg Universitet

AALBORG UNIVERSITY
DENMARK

Relational Fusion Networks

Graph Convolutional Networks for Road Networks

Skovgaard Jepsen, Tobias; Jensen, Christian S.; Nielsen, Thomas Dyhre

Published in:
IEEE Transactions on Intelligent Transportation Systems

DOI (link to publication from Publisher):
10.1109/TITS.2020.3011799

Publication date:
2022

Document Version
Early version, also known as pre-print

Link to publication from Aalborg University

Citation for published version (APA):
Skovgaard Jepsen, T., Jensen, C. S., & Nielsen, T. D. (2022). Relational Fusion Networks: Graph Convolutional Networks for Road Networks. *IEEE Transactions on Intelligent Transportation Systems*, *23*(1), 418-429. Article 9167450. Advance online publication. https://doi.org/10.1109/TITS.2020.3011799

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal -

Take down policy
If you believe that this document breaches copyright please contact us at vbn@aub.aau.dk providing details, and we will remove access to the work immediately and investigate your claim.

Downloaded from vbn.aau.dk on: April 18, 2024

Relational Fusion Networks: Graph Convolutional Networks for Road Networks

Tobias Skovgaard Jepsen, Christian S. Jensen, and Thomas Dyhre Nielsen
Department of Computer Science, Aalborg University, Denmark

***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.

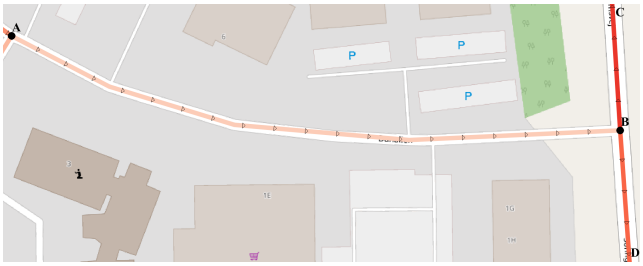


Fig. 1. Two three-way intersections in Denmark. We illustrate the observed driving speeds for one driving direction per segment and color them accordingly: the darker, the faster the speed. Black dots mark intersections and triangles indicate driving directions.

I. 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*.

As an example, the three-way intersection to the right in Fig. 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, Relational Fusion Networks (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 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 V, we review related work. In Section II, we give the necessary background on graph modeling of road networks and GCNs. In Section III, we describe RFNs in detail. In Section IV, we describe our experiments and present our results. Finally, we conclude in Section VI.

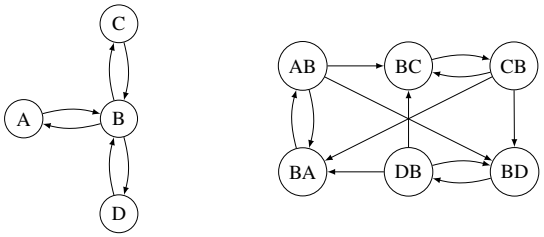
II. Preliminaries

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

A. 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 Fig. 1 is shown in Fig. 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.



(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 Fig. 1.

B. 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.

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

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 Eq. 1.

The Aggregate function in Eq. 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]:

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

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].

III. 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.

A. 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.

B. Method Overview

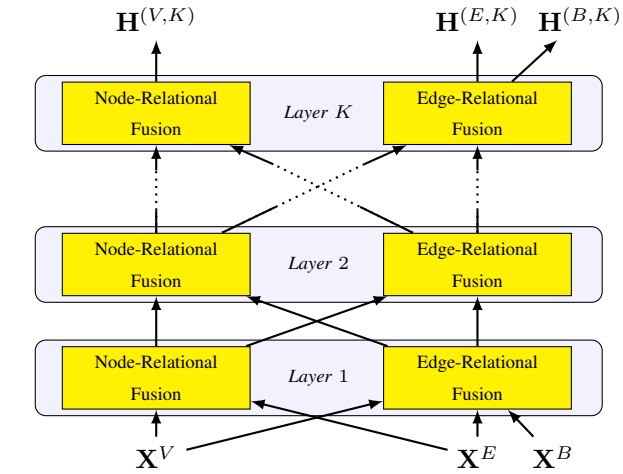
Fig. 3a gives an overview of our method. As shown, an RFN consists of K relational fusion layers, where $K \geq 1$. It takes as input feature matrices $\mathbf{X}^V \in \mathbb{R}^{|V| \times d_V}$, $\mathbf{X}^E \in \mathbb{R}^{|E| \times d_E}$, and $\mathbf{X}^B \in \mathbb{R}^{|B| \times d_B}$ that 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 III-C. 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.

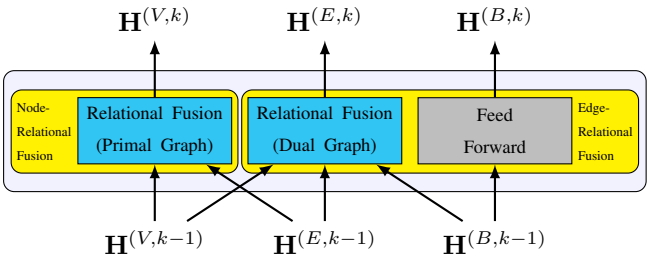
Edge-relational fusion is performed similarly to node-relational fusion but is applied on the dual graph representation. Recall, that in the edge-relational view, the relation between two edges (u, v) and (v, w) is in part described by their between-edge attributes, but also by the node attributes of v that describes the characteristics of the intersection between them. Thus, as illustrated in Fig. 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 Fig. 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.

Fig. 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 $\mathbf{H}^{(B,k)}$.



(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.

C. Relational Fusion

We present the pseudocode for the relational fusion operator at the kth layer in Algorithm 1. 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*.

In Algorithm 1, the relational representations at layer k are computed by a fusion function $\textsc{Fuse}^k$ that outputs $d_{(F,k)}$-dimensional relational representations. For each relation (v, n), $\textsc{Fuse}^k$ 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

Algorithm 1 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: **end for**
8: **return** $\mathbf{H}^{(V,k)}$
9: **end function**

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 AGGREGATE^k.

D. 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:

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

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). Eq. 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:

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

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 ($1 \le i \le d_R$) value of vector $\mathbf{H}^{(I,k)}$ is

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

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.

E. 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.

a) Attentional Aggregator: The attentional aggregator we propose computes a weighted mean over the relational representations. Formally, the attentional aggregator computes the AGGREGATE in Algorithm 1 as

$$\text{AGGREGATE}(F_v) = \sum_{\mathbf{f}_n \in F_v} A(v,n)\mathbf{f}_n, \quad (6)$$

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

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

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

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

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 Eq. 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.

F. Forward Propagation

Algorithm 2 Forward Propagation Algorithm

```
1: function ForwardPropagation(X^V, X^E, X^B)
2:     let H^(V,0) = X^V, H^(E,0) = X^E, and H^(B,0) = X^B
3:     for k = 1 to K do
4:         H^(V,k) ← RelationalFusion^k(G^P, H^(V,k-1), H^(E,k-1))
5:         H^(B',k-1) ← Join(H^(V,k-1), H^(B,k-1))
6:         H^(E,k) ← RelationalFusion^k(G^D, H^(E,k-1), H^(B',k-1))
7:         H^(B,k) ← FeedForward^k(H^(B,k-1))
8:     end for
9:     return H^(V,K), H^(E,K), H^(B,K)
10: end function
11: function Join(H^V, H^E)
12:     for all b ∈ B do
13:         let b = ((u, v), (v, w))
14:         H_b^B' ← H_b^B ⊕ H_v^V
15:     end for
16:     return H^B'
17: end function
```

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 $k = K$.

TABLE I
ROAD NETWORK SIZES.

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

IV. 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.

A. Data Set

We extract the spatial representation of the Danish municipalities of Aalborg (AAL), Brønderslev (BRS), and Copenhagen (CPH) from OpenStreetMap (OSM) [14] and convert them to their primal and dual graph representations as described in Section II. The sizes of these networks can be seen in Table I. 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

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

TABLE II
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

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 II shows the total number of driving speeds and speed limits with the proportion between training, validation, and testing sets.

B. 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 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) label 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.

C. 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].

a) 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.

D. 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 III. When reading Table III, 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

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

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

TABLE III
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	$\mathbf{6.797 \pm 0.124}$	$\mathbf{8.587 \pm 0.238}$	$\mathbf{0.535 \pm 0.014}$

results shown in Table III. In addition, none of the models give reasonable results on cross-network speed limit classification where top-performance across all models is reduced to almost a third. Hence, we also exclude these results from Table III.

1) Within-Network Inference: As shown in Table III, 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 III 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 I of the problems with direct inheritance during neighborhood aggregation in the context of road networks.

2) Cross-Network Inference: Table III 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.

E. Case Study: Danalien

Table III 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 Fig. 1. In this study, we consider a larger part of the residential area, shown in Fig. 4. 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-distributed Stochastic Neighbor Embedding (t-SNE) [20] and normalizing the dimensions to be between 0 and 1. These representations are visualized in Fig. 5.

The segments AB, BA, BC, and CB in Fig. 4 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 Fig. 5.

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 Fig. 5a, the road segments are grouped with road segments of the same category. As shown in Fig. 5b, 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 Fig. 5, the other RFN variants have a similar effect on the position of the road segments in the representation space.

As shown in Fig. 5c, the GraphSAGE model tightens the groups from Fig. 5a 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.

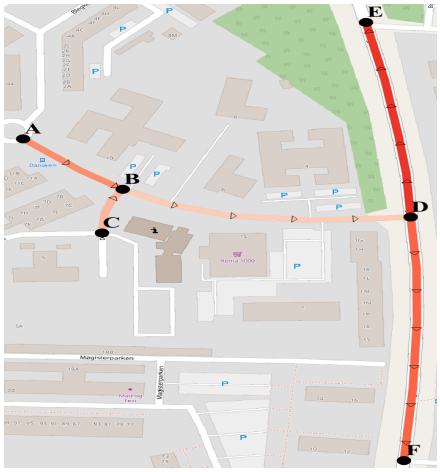


Fig. 4. Ground truth driving speeds of the Danalien region in Aalborg, Denmark. Triangles indicate direction, black dots mark nodes, and color indicates speed: the darker, the faster.

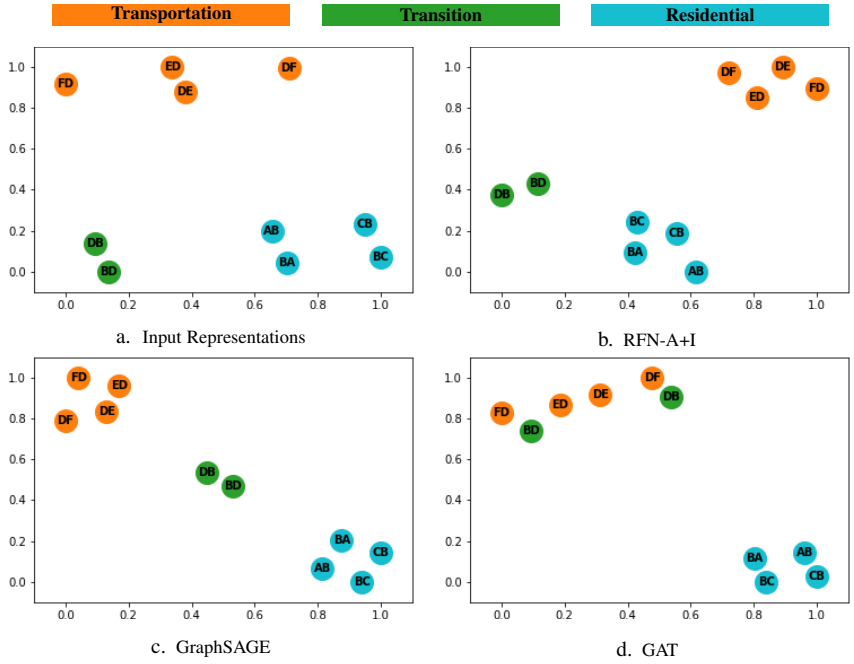


Fig. 5. t-SNE visualizations of the edge representations in the Danalien case (a) before and (b, c, d) after the first graph convolution in different models.

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 Fig. 5d, 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 Fig. 5d, 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.

V. 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.

VI. 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.

A. 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 V, 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.

ACKNOWLEDGMENTS

The research presented in this paper is supported in part by the DiCyPS (Center for Data-Intensive Cyber-Physical Systems) project and by grants from the Obel Family Foundation and VILLUM FONDEN. We also thank the OpenStreetMap (OSM) contributors who have made this work possible. Map data is copyrighted by the OpenStreetMap contributors and is available from https://www.openstreetmap.org.

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.

[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.

[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.

[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.

[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.

[20] L. van der Maaten, "Accelerating t-SNE using Tree-Based Algorithms," *JMLR*, 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.

[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.

Tobias Skovgaard Jepsen is Ph.D. Fellow in Computer Science at Aalborg University, Denmark. He received his B.S. degree in software engineering in 2015 and his M.S. degree in computer science in 2017 from Aalborg University. His research interests concern the design and application of machine learning methods for transportation tasks.

Christian S. Jensen is Professor of Computer Science at Aalborg University, Denmark. He was recently at Aarhus University for three years and at Google Inc. for one year. His research concerns data analytics and data-intensive systems, and its focus is on temporal and spatiotemporal analytics. Christian is an ACM and an IEEE fellow, and he is a member of the Academia Europaea, the Royal Danish Academy of Sciences and Letters, and the Danish Academy of Technical Sciences.

Thomas Dyhre Nielsen is Professor MSO (with special obligations) at Aalborg University, Denmark. His research interests concern different aspects of decision support systems, but with particular focus on machine learning and probabilistic graphical models including Bayesian networks. He is a co-author on a textbook on probabilistic graphical models and decision analysis and he is senior area editor for the International Journal of Approximate Reasoning.

APPENDIX 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. Node Attributes

We derive node attributes using the zone map from the Danish Business Authority mentioned in Section IV-A. 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.

B. 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.

C. 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.

APPENDIX B
HYPERPARAMETER SELECTION

As mentioned in Section IV-C, 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.

A. 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. 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.

C. 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.

D. Selected Hyperparameters

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

TABLE IV
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	$d = 32$	$\lambda = 0.01$		$d = 64$	$\lambda = 0.01$	

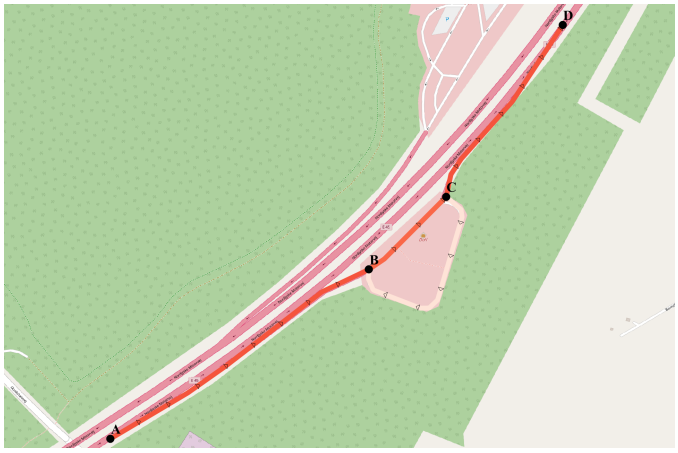


Fig. 6. Ground truth driving speeds of a resting area and its adjacent motorway segments near the village of Dall, Denmark. Triangles indicate direction, black dots mark nodes, and color indicates speed: the darker, the faster.

APPENDIX C
CASE STUDY: DALL

The Danalien case discussed in Section IV-E 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 I, extreme cases of volatile homophily exists where there is no homophily. Such a case is illustrated in Fig. 6 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 Fig. 6, segment CB has a substantially different driving speed than the other segments.

We 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 Fig. 7 following the same procedure as in the Danalien case (cf. Section IV-E).

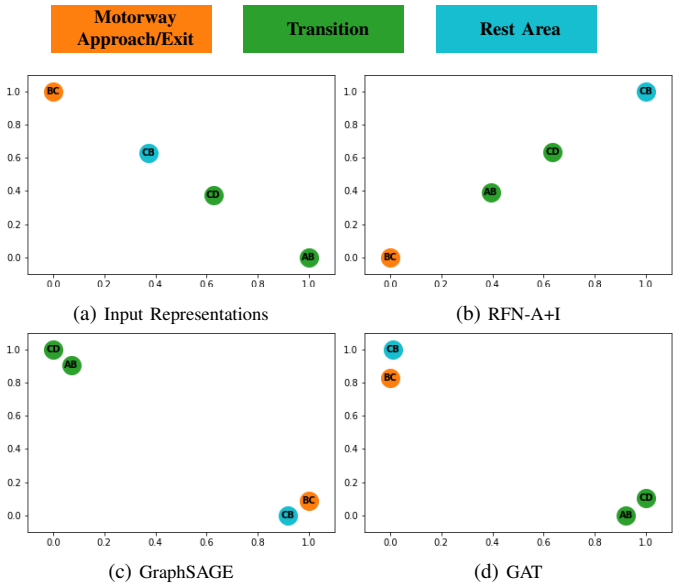


Fig. 7. t-SNE visualizations of the edge representations in the Dall case (a) before and (b, c, d) after the first graph convolution in different models.