Traffic Accident Prediction using Graph Neural Networks: New Datasets and the TRAVEL Model

Baixiang Huang
huangb@comp.nus.edu.sg
National University of Singapore
Singapore

Bryan Hooi
bhooi@comp.nus.edu.sg
National University of Singapore
Singapore

ABSTRACT

Traffic accident prediction is crucial for reducing and mitigating road traffic accidents. Many existing machine learning approaches predict the number of traffic accidents in each cell of a discretized grid without considering the underlying graph structure of road networks. To allow us to incorporate road network information, graph-based approaches such as Graph Neural Networks (GNNs) are a natural choice. However, applying GNNs to the accident prediction problem is made challenging by a lack of suitable graph-structured traffic accident prediction datasets. To overcome this problem, we first construct one thousand real-world graph-based traffic accident datasets, along with two benchmark tasks (accident occurrence prediction and accident severity prediction). We then comprehensively evaluate eleven state-of-the-art GNN variants using the created datasets. Moreover, we propose a novel Traffic Accident Vulnerability Estimation via Linkage (TRAVEL) model, which is designed to capture angular and directional information from road networks. We demonstrate that the TRAVEL model consistently outperforms the GNN baselines. The datasets and code are available at https://github.com/baixianghuang/travel.

KEYWORDS

datasets, graph neural networks, traffic accident prediction

1 INTRODUCTION

Road traffic accidents are a major global public health issue, and are the leading cause of death for people aged five to twenty-nine years globally [24]. In this work, we focus on environmental risk factors of road traffic accidents - in particular, can we predict how risk-prone a traffic intersection is: that is, the number and severity of accidents occurring near the intersection, based only on readily available street map data, such as the road geometry, nearby highways, landmarks, etc.? This work can help governments to mitigate traffic risks, such as by informing the design of future road networks, and in the planning of accident response systems with the awareness of risk-prone accident hotspots.

A city road network has a variety of road features that could potentially be correlated with accidents - such as road type, road length, and the number of lanes. An analysis of traffic accidents in the United States between 2016 and 2020 shows that most of the accidents took place near junctions or intersections [15]. Hence, our goal is to design an algorithm that learns how this information should be used to predict the riskiness of each road intersection.

Existing studies applying machine learning to traffic accident prediction typically use non-graph-based approaches: most commonly, they discretize the data based on a spatial grid, then use various features of each grid cell to predict the number of accidents in each cell. However, this approach has the drawback of relying on manually designed features, rather than the more flexible approach of allowing the algorithm to learn directly from real-world street map data.

In contrast, our approach treats the map data as a graph, allowing us to flexibly incorporate road features from both a location itself and its nearby locations: for example, the shape or angle of roads leading to an intersection. Moreover, graph-based approaches allow much longer-range information to be taken into account by aggregating information from neighbors multiple hops away, in contrast to existing grid-based approaches which only use information from within each grid cell.

In addition, using graph-based approaches allows us to take advantage of rich and readily available sources of geospatial data such as OpenStreetMap (OSM). Maps from OSM are already structured as graphs, where nodes represent intersections and dead-end nodes, and edges represent roads. This allows our approach to be applied for predicting risky intersections wherever map data is available, without having to collect or digitize additional customized datasets for a specific area.

To learn from graph data, Graph Neural Networks (GNNs) are becoming increasingly popular [26]. However, it is difficult to apply GNNs to the traffic accident prediction problem. One major reason is the lack of suitable datasets: the current accident datasets are not organized alongside a graph structure. To overcome this problem, we construct a new set of datasets for one thousand US cities, with two prediction tasks (accident occurrence prediction and accident severity prediction). We do this by merging and preprocessing data from the US-Accidents dataset [15] with graph-based street geospatial data from OpenStreetMap. These datasets allow users to apply graph-based approaches to the traffic accident prediction problem, without significant preprocessing effort.

Moreover, we develop a novel GNN architecture called Traffic Accident Vulnerability Estimation via Linkage (TRAVEL). In contrast to existing GNNs, where each node aggregates from its neighbors using simple functions like 'mean' or 'sum', ours aggregates in a way that captures both the angles and directions of roads adjacent to a node. TRAVEL's graph convolution layers consist of two components: the first component captures the angles between roads, while the second component aims to learn the direction information of the roads.

The key contributions of this work are as follows: (1) We formulate the graph-based traffic accident prediction as a node prediction problem, which aims to predict accident occurrences or accident severities over a road graph. (2) We construct and release one thousand benchmark datasets with two accident-related prediction tasks to enable the use of graph-based approaches for accident

prediction, by merging real-world graph information from OpenStreetMap with accident data from the US-Accidents dataset [15]. (3) We propose a new GNN architecture, TRAVEL, which can capture angular and directional information from road networks. We comprehensively evaluate our model against MLP, XGBoost, and eleven existing GNN baselines. We validate that our proposed model consistently has the best performance on the benchmark datasets.

2 RELATED WORK

The traffic accident prediction problem is often formulated as a spatiotemporal forecasting task. Early studies apply k-nearest neighbors, k-means clustering, and logistic regression [14, 19]. Chen et al. [4] use human mobility features obtained from stacked denoising autoencoders to infer traffic risk. Yu et al. [28] combine Long Short-Term Memory (LSTM) and stacked autoencoders for post-accident condition prediction. Zhou et al. [30] use deep learning for spatiotemporal accident prediction over a rectangular grid. With the emergence of graph-based deep learning approaches, some work applies GNNs to spatiotemporal traffic accident forecasting [27, 29, 31].

Traffic accident prediction has been also formulated as a classification or regression problem without considering temporal information. Early work proposed Poisson, Negative Binomial, and Negative Multinomial regression models to predict the number of accidents over a discretized grid [3, 17]. Najjar et al. [16] train traditional CNNs on satellite images of traffic accidents to produce a traffic risk map. These studies convert road networks to regular 2-D grids because traditional convolutional operations handle spatial correlations over such grids. Also, existing work generally does not use street map data. In contrast, our approach not only takes full advantage of graph structures of road networks but also incorporates real-world environmental features. Accident timestamps are included in our proposed datasets but are not used in our experiments because we focus on features unrelated to time.

3 PRELIMINARIES

3.1 Problem Statement

We model a road network as a weighted directed graph $G = (V, E)$, where vertices in V represent endpoints (intersections and dead-end nodes) in the road network, while edges in E represent roads. The graph also has a set of node features $\mathbf{x}_v \in \mathbb{R}^{d_v}$, where d_v is the number of node features, and edge attributes $\mathbf{e}_{uv} \in \mathbb{R}^{d_e}$ for $(u, v) \in E$, where d_e is the number of edge attributes.

In the 'accident occurrence' prediction task, we aim to output a binary prediction indicating whether a vertex has traffic accidents or not. In the 'accident severity' prediction task, we bucketize the average severity of accidents at each node, thereby grouping the nodes into a finite number of classes. Then, we aim to predict the class of each node. Therefore, both of these tasks are formulated as node classification problems. Each node v has its label denoted as y_v.

3.2 Basic GNN Framework

A GNN model uses node features $\mathbf{x}_v$ to learn the representation of a node. In each layer, each node aggregates representations of its neighbors and updates the representation of itself. The k-th layer of a basic GNN is:

$$\mathbf{h}_v^{(k)} = \mathsf{Update}^{(k)}(\mathbf{h}_v^{(k-1)}, \mathbf{m}_{\mathcal{N}(v)}^{(k)}) \tag{1}$$

$$\mathbf{m}_{\mathcal{N}(v)}^{(k)} = \mathsf{Aggr}^{(k)}(\{\mathbf{h}_u^{(k-1)}, \forall u \in \mathcal{N}(v)\}) \tag{2}$$

where $\mathbf{h}_v^{(k)}$ is the node embedding of node v in the k-th layer. Initial 0-th layer embeddings $\mathbf{h}_v^{(0)}$ are equal to node features $\mathbf{x}_v$. $\mathbf{m}_{\mathcal{N}(v)}$ denotes the aggregated message from node v's neighborhood $\mathcal{N}(v) = \{u : (u, v) \in E\}$. Update is a neural network that updates the representation of v, and Aggr is a function that aggregates representations of $\mathcal{N}(v)$.

Table 1: Statistics of one thousand datasets.

Measure	# nodes	# edges	Avg node degree	% accident nodes
mean	3075.47	7845.19	2.55	10.23
std	5581.99	14313.71	0.18	8.58
min	45.00	106.00	1.73	1.05
25%	672.00	1708.75	2.42	5.11
50%	1487.00	3638.00	2.53	7.58
75%	3049.25	7752.75	2.66	11.83
max	59694.00	149281.00	3.17	62.18

4 DATASET CONSTRUCTION

A major obstacle explaining why it is difficult to apply GNNs to traffic accident prediction is the lack of suitable graph-based datasets. Hence, in this section, we describe how we construct and release one thousand Traffic Accident Benchmark (TAB-1k) datasets by combining the US-Accidents dataset [15] and road data from OpenStreetMap (OSM). Numerical measures of TAB-1k can be found in Table 1. To provide a convenient and user-friendly interface, we release our datasets following the format of datasets in PyTorch Geometric [8], a popular package for GNNs. Initializing our datasets will automatically download the preprocessed files, the result of which can be directly plugged into existing GNNs.

4.1 Data Collection

The raw accident events come from the US-Accidents dataset [15], a real-world traffic accident dataset that covers forty-nine states of the United States. It contains more than 4.23 million instances of traffic accidents data between February 2016 and December 2020. Moosavi et al. [15] report that roughly 32% of accidents occurred on or near local roads (e.g., streets, avenues, and boulevards), and about 40% took place on or near high-speed roads (e.g., highways, interstates, and state roads). Their analyses also demonstrate that most of the accidents took place near intersections. We use the data across the entire four-year time period. Based on the data from the US-Accident dataset, we sort the cities by their total counts of traffic accident occurrences. Next, we select one thousand cities with the most accidents to construct our datasets.

To construct our graphs, we combine the accident data with geospatial data from OpenStreetMap, a collaborative initiative that

Table 2: Edge features (top) and node features (bottom) included in our datasets.

Graph feature	Description
highway	The type of a road (tertiary, motorway, etc.).
length	The length of a road.
bridge	Indicates whether a road represents a bridge.
lanes	The number of lanes of a road.
oneway	Indicates whether a road is a one-way street.
maxspeed	The maximum legal speed limit of a road.
access	Describes restrictions on the use of a road.
tunnel	Indicates whether a road runs in a tunnel.
junction	Describes the junction type of a road.
highway	The road type of a node.
street_count	The number of roads connected to a node.

provides a freely available geospatial database. The OSM data contain rich environmental features such as road type, road length, bridge type, and the number of lanes. These features are also keys for OSM tags, which describe the specific attributes of map elements (nodes, ways, or relations). We use the OSMnx [2] package to download geospatial data from OSM to build traffic road networks. OSM includes walkable, drivable, and bikeable urban road data. Since most accidents occur on the drivable networks, we only use drivable public road data (private-access or service roads not included). In a road network, nodes are points such as intersections and dead-ends, and edges represent roads.

4.2 Data Preprocessing

We first build road networks using structural and feature information from OSM. The edge and node features, and their descriptions, are listed in Table 2. Next, missing values are replaced with a new category, and feature data are encoded using one-hot encoding. Then the coordinate data of accident locations from the US-Accidents dataset are used to find the nearest corresponding nodes in the road networks based on the haversine distance. For the accident occurrence prediction task, binary labels are added to each node indicating whether it contains at least one accident. For the severity prediction task, average accident severities are bucketized into eight classes (using an interval size of 0.5) to be used as labels. Finally, data are split using a stratified split: 60% of the data is used for training, 20% is used for validation, and the remaining 20% is used for testing.

5 TRAVEL FRAMEWORK

5.1 Overview and Motivation

Road geometry-related characteristics such as turning radius and direction (i.e. left versus right turns) have long been recognized as important factors affecting road safety [18]. Motivated by this, we design a GNN approach that is effective at capturing information from both road geometry, as well as allowing the use of rich node and edge features already available in OSM.

How do we design a GNN architecture that effectively incorporates road geometry? We find that an effective way to do this is to augment the message passing process in GNNs with additional *angular* and *directional* information. The angular component allows our model to better capture relevant information about an intersection (e.g. whether it has a right or left turn, sharp turns, etc.). The directional component allows the model to capture the direction of a road: e.g. whether it is heading north-to-south versus east-to-west, which can be relevant in practice.

We describe TRAVEL as a layer taking in the previous node embeddings $\mathbf{h}_v$ (suppressing the layer number, since we only describe a single TRAVEL layer).

5.2 Angular Component

The angular component augments the message passing process from node u to node v with information about the angles between the road (u, v) and all the other roads intersecting at node v. This allows our GNN model to take into consideration the angles between roads, which are of key importance to road geometry, throughout the message passing process.

An important aspect of the design of our angular component is that it is *rotationally symmetric*. This means that the output computed by this component at a node v does not change even if we rotate the graph by any rotation centered at v. This makes sense intuitively: the road geometry of a particular intersection does not change if we rotate all roads about this intersection by any fixed angle. This rotational symmetry is important as it ensures that this component has *inductive biases*[1] that are well suited to its intended task of capturing road geometry.

Given points u, v, w, let $\angle(\overrightarrow{uv}, \overrightarrow{wv})$ denote the directed angle[2] from $\overrightarrow{uv}$ to $\overrightarrow{wv}$. Recall that our angular component is designed to augment the messages passed from u to v with information about the angles between road (u, v) and other roads intersecting at v.

Formally, the *set of (directed) angles* between road (u, v) and each of the other roads at v is:

$$\Phi_{uv} := \{\angle(\overrightarrow{uv}, \overrightarrow{wv}) : w \in \mathcal{N}_v \setminus \{u\}\} \tag{3}$$

Next, we will aggregate over Φ_{uv} to extract suitably summarized information from this set. Rather than using standard aggregation functions, we use a designed aggregation function that aims to particularly emphasize the presence of informative features: namely 1) sharp left turns, 2) sharp right turns, and 3) nearly straight roads. We do this by first defining Φ^{π}_{uv} as the set $\{|\pi - \phi| : \phi \in \Phi_{uv}\}$, then aggregating as follows (where $\|$ denotes concatenation):

$$\mathbf{a}_{uv} := \min(\Phi_{uv}) \,\|\, \max(\Phi_{uv}) \,\|\, \min(\Phi^{\pi}_{uv}) \tag{4}$$

The first two components correspond to the sharpest angles of left and right turns to edge (u, v). The third component corresponds to the angle of (u, v) to the road which is closest to a straight road along with (u, v). Thus, the aggregated angular information $\mathbf{a}_{uv}$ provides a concise summary of the useful information contained in angles between (u, v) and other roads at v.

Finally, our angular component incorporates this angular information $\mathbf{a}_{uv}$ when passing a message along (u, v). The angular

[1] Inductive biases describe the assumptions that a model uses to produce output on unseen data.

[2] This is the signed angle that $\overrightarrow{uv}$ must be rotated to have the same direction as $\overrightarrow{wv}$.

component takes in node representations $\mathbf{h}_v$, and outputs the angular node representations $\mathbf{h}_v^{\text{Angle}}$:

$$\mathbf{h}_v^{\text{Angle}} = \text{ReLU}(\mathbf{W}\mathbf{h}_v + \mathbf{m}_{\mathcal{N}(v)}^{\text{Angle}}) \tag{5}$$

$$\mathbf{m}_{\mathcal{N}(v)}^{\text{Angle}} = \sum_{u \in \mathcal{N}(v)} \text{MLP}(\mathbf{h}_u \parallel \mathbf{e}_{uv} \parallel \mathbf{a}_{uv}) \tag{6}$$

An important property of the angular component is that it is rotationally symmetric. We show this as follows.

Theorem 5.1 (Rotational Symmetry). *The angular component is rotationally symmetric.*

Proof. When rotating all points about v by a fixed rotation, each of the directed angles $\angle(\overrightarrow{uv}, \overrightarrow{wv})$ remains unchanged as u and w rotate by the same angle around v. This implies that all the $\mathbf{a}_{uv}$ also remain unchanged. Since the edge features $\mathbf{e}_{uv}$ are also unchanged by the rotation, thus $\mathbf{h}_v^{\text{Angle}}$ is also unchanged. □

5.3 Directional Component

As we have seen, the angular component is designed to be rotationally symmetric for the purpose of modeling road geometry. However, the directional information ignored by the angular component can still be useful in some contexts: e.g. in some cities, north-south roads may have different characteristics from east-west roads. This motivates our directional component, which captures the direction that each road is heading in.

Let LAT_u and LON_u denote the latitude and longitude of node u, respectively. This allows us to compute the **direction** of the edge (u, v) as:

$$\mathbf{d}_{uv} = (\text{LAT}_v - \text{LAT}_u, \text{LON}_v - \text{LON}_u) \tag{7}$$

Like in the angular component, we incorporate directions into the message passing process:

$$\mathbf{h}_v^{\text{Dir}} = \text{ReLU}(\mathbf{W}\mathbf{h}_v + \mathbf{m}_{\mathcal{N}(v)}^{\text{Dir}}) \tag{8}$$

$$\mathbf{m}_{\mathcal{N}(v)}^{\text{Dir}} = \sum_{u \in \mathcal{N}(v)} \text{MLP}(\mathbf{h}_u \parallel \mathbf{e}_{uv} \parallel \mathbf{d}_{uv}) \tag{9}$$

5.4 Combined TRAVEL Layer

Finally, the combined TRAVEL layer's output is simply the concatenation between the output of the angular and directional components, i.e. $\mathbf{h}_v^{\text{Angle}} \parallel \mathbf{h}_v^{\text{Dir}}$. This TRAVEL layer can be straightforwardly trained using standard loss functions (cross-entropy loss in our setting), or plugged into any existing GNN. The angular and directional features are provided in the released datasets.

6 EXPERIMENTS

6.1 Traffic Accident Occurrence Prediction

In the accident occurrence prediction task, which is formulated as a node classification problem, we aim to predict whether a node has traffic accidents or not based on previous accident records. Table 4 and Figure 3 from Appendix A show the prediction results on the created datasets. We generally observe that: (1) The proposed TRAVEL consistently achieves the best performance on all the metrics, due to its ability to capture angular and directional features on top of other environmental features. (2) GNN-based approaches generally outperform XGBoost and MLP. This is because nodes in GNNs can aggregate feature information from their neighbors, while the MLP and XGBoost models can only learn from local feature data. (3) GNN variants that support multi-dimensional edge features generally outperform models that do not support them.

6.2 Traffic Accident Severity Prediction

The goal of this task is to predict the severity of accidents. Since a node may have multiple accidents with different severity, we compute each node's mean severity and bucketize this mean severity into eight classes. Accident severity is represented by a number between 0 and 7, where 0 denotes no accident, 1 indicates the most negligible impact on traffic, and 7 indicates a significant impact on traffic. Experiment results (Table 5 and and Figure 4) provided in Appendix A demonstrate that the TRAVEL model again clearly outperforms the baselines across all datasets in terms of weighted F1 score.

Further details about software used, architectures, and hyperparameters, as well as hyperparameter robustness and running time experiments, can be found in Appendix A. On the whole, they show that TRAVEL performs well even with very minimal hyperparameter tuning, and its running time is comparable to other GNNs (particularly those that use edge features), and slightly faster than MPNNs on most datasets.

7 CONCLUSION

In this paper, we affirm the benefits of GNNs for the critically important task of identifying risky road intersections. We first formulate the accident occurrence prediction and accident severity prediction tasks as graph-based node classification problems. To stimulate future research in this area, we construct and release one thousand graph-based Traffic Accident Benchmark datasets (TAB-1k) with these two benchmark tasks, and evaluate thirteen state-of-the-art machine learning approaches on them. Furthermore, we propose our TRAVEL framework, designed to capture angular and directional attributes, and prove its theoretical property (rotational symmetry of angular component). The experiments show that TRAVEL consistently outperforms the baselines. Practically, TRAVEL requires only features available in OpenStreetMaps as input, so it can be readily applied to almost all cities. For future work, we plan to incorporate larger state-level networks and investigate extensions such as model explainability.

REFERENCES

[1] Filippo Maria Bianchi, Daniele Grattarola, Lorenzo Livi, and Cesare Alippi. 2021. Graph Neural Networks with Convolutional ARMA Filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2021), 1–1. https://doi.org/10.1109/tpami.2021.3054830

[2] Geoff Boeing. 2017. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems* 65 (2017), 126–139.

[3] Ciro Caliendo, Maurizio Guida, and Alessandra Parisi. 2007. A crash-prediction model for multilane roads. *Accident Analysis & Prevention* 39, 4 (2007), 657–670.

[4] Quanjun Chen, Xuan Song, Harutoshi Yamada, and Ryosuke Shibasaki. 2016. Learning deep representation from big and heterogeneous data for traffic accident inference. In *Thirtieth AAAI conference on artificial intelligence*.

[5] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[6] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems* 29 (2016), 3844–3852.
[7] Jian Du, Shanghang Zhang, Guanhang Wu, Jose M. F. Moura, and Soummya Kar. 2018. Topology Adaptive Graph Convolutional Networks. arXiv:1710.10370 [cs.LG]
[8] Matthias Fey and Jan Eric Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).
[9] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.
[10] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.
[11] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery* 33, 4 (2019), 917–963.
[12] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).
[13] Guohao Li, Chenxin Xiong, Ali Thabet, and Bernard Ghanem. 2020. DeeperGCN: All You Need to Train Deeper GCNs. arXiv:2006.07739 [cs.LG]
[14] Yisheng Lv, Shuming Tang, and Hongxia Zhao. 2009. Real-time highway traffic accident prediction based on the k-nearest neighbor method. In *2009 international conference on measuring technology and mechatronics automation*, Vol. 3. IEEE, 547–550.
[15] Sobhan Moosavi, Mohammad Hossein Samavatian, Srinivasan Parthasarathy, Radu Teodorescu, and Rajiv Ramnath. 2019. Accident risk prediction based on heterogeneous sparse data: New dataset and insights. In *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 33–42.
[16] Alameen Najjar, Shun'ichi Kaneko, and Yoshikazu Miyanaga. 2017. Combining satellite imagery and open data to map road safety. In *Thirty-First AAAI Conference on Artificial Intelligence*.
[17] Jutaek Oh, Simon P Washington, and Doohee Nam. 2006. Accident prediction model for railway-highway interfaces. *Accident Analysis & Prevention* 38, 2 (2006), 346–356.
[18] Sarbaz Othman, Robert Thomson, and Gunnar Lannér. 2009. Identifying critical road geometry parameters affecting crash rate and crash type. In *Annals of Advances in Automotive Medicine/Annual Scientific Conference*, Vol. 53. Association for the Advancement of Automotive Medicine, 155.
[19] Seong-hun Park, Sung-min Kim, and Young-guk Ha. 2016. Highway traffic accident prediction using VDS big data analysis. *The Journal of Supercomputing* 72, 7 (2016), 2815–2831.
[20] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019), 8026–8037.
[21] Yunsheng Shi, Zhengjie Huang, Shikun Feng, Hui Zhong, Wenjin Wang, and Yu Sun. 2021. Masked Label Prediction: Unified Message Passing Model for Semi-Supervised Classification. arXiv:2009.03509 [cs.LG]
[22] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).
[23] Boris Weisfeiler and Andrei Leman. 1968. The reduction of a graph to canonical form and the algebra which appears therein. *NTI, Series* 2, 9 (1968), 12–16.
[24] WHO et al. 2018. *Global status report on road safety 2018: summary*. Technical Report. World Health Organization.
[25] Tian Xie and Jeffrey C. Grossman. 2018. Crystal Graph Convolutional Neural Networks for an Accurate and Interpretable Prediction of Material Properties. *Phys. Rev. Lett.* 120 (Apr 2018), 145301. Issue 14. https://doi.org/10.1103/PhysRevLett.120.145301
[26] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).
[27] Le Yu, Bowen Du, Xiao Hu, Leilei Sun, Liangzhe Han, and Weifeng Lv. 2021. Deep spatio-temporal graph convolutional network for traffic accident prediction. *Neurocomputing* 423 (2021), 135–147.
[28] Rose Yu, Yaguang Li, Cyrus Shahabi, Ugur Demiryurek, and Yan Liu. 2017. Deep learning: A generic approach for extreme condition traffic forecasting. In *Proceedings of the 2017 SIAM international Conference on Data Mining*. SIAM, 777–785.
[29] Yang Zhang, Xiangyu Dong, Lanyu Shang, Daniel Zhang, and Dong Wang. 2020. A multi-modal graph neural network approach to traffic risk forecasting in smart urban sensing. In *2020 17th Annual IEEE international conference on sensing, communication, and networking (SECON)*. IEEE, 1–9.
[30] Zhengyang Zhou, Yang Wang, Xike Xie, Lianliang Chen, and Hengchang Liu. 2020. RiskOracle: A Minute-Level Citywide Traffic Accident Forecasting Framework. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1258–1265.
[31] Zhengyang Zhou, Yang Wang, Xike Xie, Lianliang Chen, and Chaochao Zhu. 2020. Foresee urban sparse traffic accidents: A spatiotemporal multi-granularity perspective. *IEEE Transactions on Knowledge and Data Engineering* (2020).

A EXPERIMENT RESULTS

In this section, we comprehensively evaluate the performance of fourteen models, finding that TRAVEL clearly outperforms the baselines on the traffic accident occurrence and severity prediction tasks. Figure 1 and 2 plot the labels to be predicted for the two tasks respectively.

A.1 Baselines

We compare our TRAVEL model with two generic machine learning models and eleven state-of-the-art GNN models: (1) **XGBoost**: A scalable gradient boosting tree-based approach [5]. (2) **MLP**: A multi-layer perceptron is a classic feedforward artificial neural network. (3) **GCN**: Graph Convolutional Networks generalize CNNs from low-dimensional regular grids to graph data using neighborhood-based filters [12]. (4) **ChebNet**: Chebyshev spectral graph convolution networks are spectral graph convolutional architectures with fast localized spectral filtering [6]. (5) **ARMANet**: Graph neural networks with convolutional auto-regressive moving average (ARMA) filters [1]. (6) **GraphSAGE**: A general inductive framework for inductive representation learning on graphs [10]. (7) **TAGCN**: Topology adaptive graph convolutional networks use fixed-size learnable filters to perform convolutions on graphs [7]. (8) **GIN**: Graph Isomorphism Networks generalize the Weisfeiler-Lehman (WL) graph isomorphism test [23, 26]. (9) **GAT**: Graph attention networks apply attentional mechanisms during aggregation [22]. (10) **MPNN**: Message Passing Neural Network is a general GNN framework designed for computational chemistry, reasoning, and simulation. [9]. (11) **CGC**: Crystal graph convolutional neural network is an accurate and interpretable framework that can extract the contributions from local features to global properties [25]. (12) **GEN**: GENeralized graph convolutional neural networks support softmax, power, and mean aggregation [13]. (13) **Transformer**: Graph transformers adopt vanilla multi-head attention into graph learning with taking into account the case of edge features [21].

Among GNN baselines, GCN, ChebNet, ARMANet, GraphSAGE, TAGCN, and GIN do not support message passing with multi-dimensional edge features. In contrast, GAT, MPNN, CGC, GEN, and Transformer support message passing with multi-dimensional edge features.

A.2 Results Analysis

Table 3 shows the statistics of eight sample datasets. It also indicates that the traffic accident data are imbalanced: only a small proportion of the nodes have accidents. Therefore, the models are evaluated based on F1 score, Area Under the Receiver Operating Characteristic Curve (AUC), and Accuracy. Table 4 and Table 5 show the prediction results on the sample datasets.

To evaluate performance across the full set of one thousand datasets, in Figure 3 and Figure 4, we use the Wilcoxon-Holm critical difference diagram[11]. It can be interpreted as follows: methods are arranged by their average rank, so the rightmost method (TRAVEL) is the one with overall best performance in terms of F1 score, i.e.

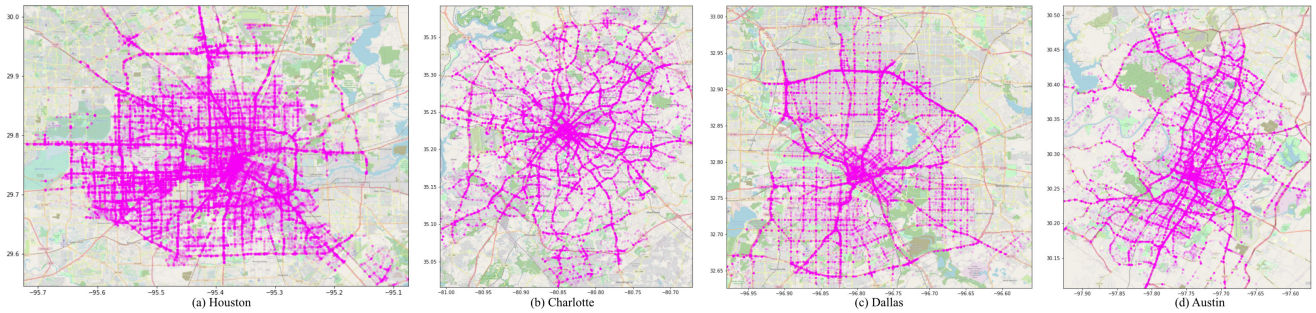

Figure 1: Traffic accident locations of Houston, Charlotte, Dallas, and Austin.

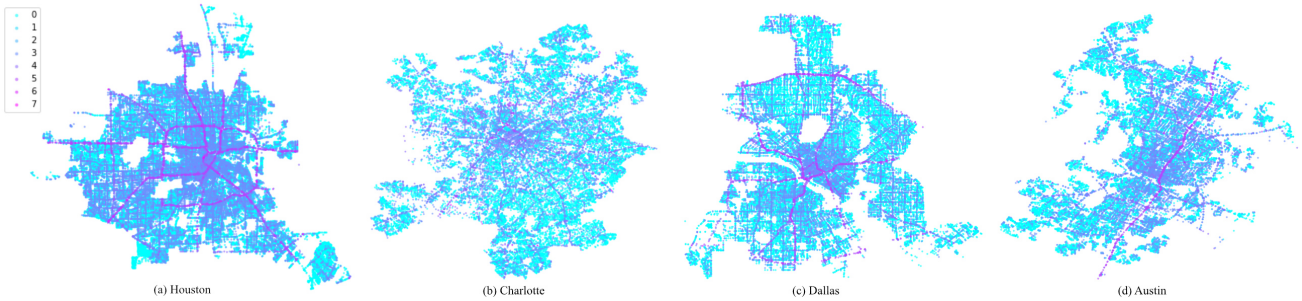

Figure 2: Traffic accident severities of Houston, Charlotte, Dallas, and Austin.

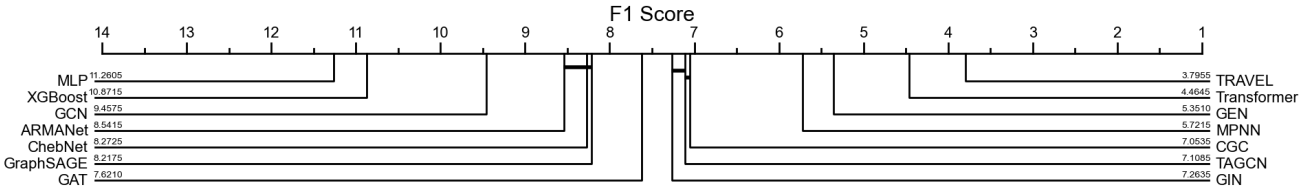

Figure 3: Critical difference diagram of the traffic accident occurrence prediction task. TRAVEL performs statistically significantly better than all other methods.

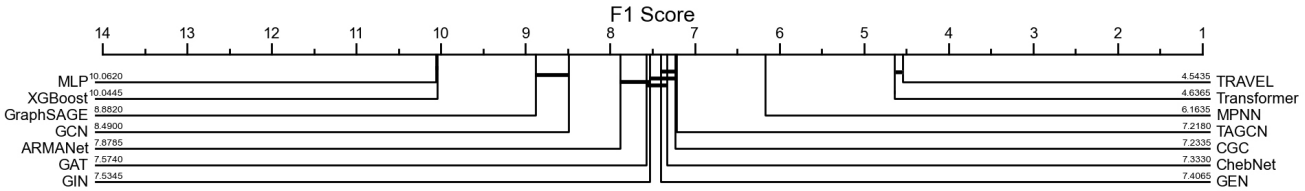

Figure 4: Critical difference diagram of the traffic accident severity prediction task. TRAVEL performs the best overall, and statistically significantly better than all other methods except Transformer.

lowest average rank. Meanwhile, the thick horizontal lines groups a set of classifiers that are not significantly different in performance from one another.

A.3 Training Time

We evaluate the training time of our TRAVEL framework against thirteen baselines. As demonstrated in Table 6, our TRAVEL model has a similar execution time compared to the MPNN model in the traffic accident occurrence task. Likewise, Table 7 shows that the running time of the TRAVEL framework is less than that of the MPNN model in most cases in the traffic accident severity prediction task.

Table 3: Statistics of eight sample datasets.

City	Houston	Charlotte	Dallas	Austin	Los Angeles	Atlanta	Seattle	Chicago
# nodes	59694	29364	36022	25549	49713	13210	19021	28668
# edges	149281	68403	92117	63554	136742	34513	50227	76242
Avg node degree	2.50	2.33	2.56	2.49	2.75	2.61	2.64	2.66
% accident nodes	33.90	31.15	30.39	35.63	12.46	14.67	26.56	15.71

Table 4: Accident occurrence prediction results in terms of F1 score(%), and AUC(%).

	Houston		Charlotte		Dallas		Austin		Los Angeles		Atlanta		Seattle		Chicago	
Classifer	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC
XGBoost	22.34	55.60	44.89	62.95	33.74	59.48	29.11	58.03	15.58	54.13	36.23	61.40	35.99	60.71	48.32	67.77
MLP	22.33	67.46	44.89	73.68	33.74	69.53	29.26	69.61	12.95	65.81	37.66	63.57	36.09	71.66	51.19	74.75
GCN	38.57	71.38	57.44	80.41	45.84	74.66	60.43	79.68	38.10	79.29	43.65	79.26	53.85	79.64	35.09	79.66
ChebNet	45.06	73.88	59.07	81.04	50.72	77.58	62.16	80.69	38.00	80.58	48.72	79.14	55.58	81.48	49.69	80.92
ARMANet	45.40	73.99	60.37	81.22	47.36	77.51	62.65	80.64	38.90	80.43	49.16	79.43	56.20	81.16	50.51	80.60
GraphSAGE	55.41	73.33	60.17	81.10	46.73	76.09	60.60	80.03	40.63	79.63	50.17	78.90	57.47	79.90	53.25	80.79
TAGCN	49.16	75.50	61.64	81.77	54.88	78.70	65.53	81.52	46.56	83.84	49.35	82.06	57.89	82.20	52.72	81.21
GIN	46.88	73.33	61.08	80.77	48.47	76.34	62.14	80.66	40.41	81.17	45.95	80.73	55.11	81.00	46.21	80.00
GAT	44.57	73.65	59.21	81.14	47.24	75.81	58.04	80.04	40.51	79.29	53.65	77.09	61.00	79.74	42.14	81.28
MPNN	59.90	79.45	69.76	84.91	55.39	81.37	68.11	84.04	45.06	83.04	53.27	86.48	66.32	85.79	50.91	85.16
CGC	58.20	78.95	68.91	85.69	52.29	80.08	67.79	83.75	45.11	81.29	52.43	86.96	65.14	85.67	51.94	83.73
Transformer	62.69	80.30	68.95	86.30	58.07	81.92	68.67	84.60	46.58	83.42	54.60	87.09	65.43	86.45	52.71	85.29
GEN	63.10	79.96	69.24	86.01	60.91	81.89	70.12	84.75	48.01	80.25	55.78	86.11	63.61	85.92	53.81	84.44
TRAVEL	**65.29**	**81.16**	**70.28**	**86.38**	**64.40**	**83.02**	**71.70**	**85.69**	**51.91**	**84.68**	**61.01**	**88.20**	**68.34**	**86.60**	**55.96**	**85.32**

Table 5: Accident severity prediction results in terms of weighted F1 score(%) and accuracy(%).

	Houston		Charlotte		Dallas		Austin		Los Angeles		Atlanta		Seattle		Chicago	
Classifier	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc
XGBoost	59.54	68.58	68.84	72.34	65.96	72.58	59.37	68.00	81.72	87.12	80.43	86.14	71.12	77.12	78.38	84.74
MLP	59.54	68.58	63.04	71.62	65.96	72.76	59.14	67.96	81.71	87.15	80.43	86.14	70.91	76.99	78.38	84.74
GCN	60.78	67.98	72.12	74.79	67.18	72.34	69.76	72.87	83.12	87.74	81.27	86.52	74.33	78.15	79.40	85.18
ChebNet	61.82	68.67	72.24	74.94	67.79	73.15	69.66	72.26	83.19	87.67	82.05	86.67	74.80	78.25	78.64	84.85
ARMANet	61.06	68.74	73.03	75.31	67.48	72.98	69.96	72.56	83.38	87.70	81.49	86.52	74.65	78.44	79.73	85.32
GraphSAGE	61.90	69.04	73.67	75.42	67.98	73.11	69.52	72.63	83.67	87.80	81.32	86.40	74.64	78.10	79.57	85.25
TAGCN	62.67	69.05	73.63	75.21	68.69	73.29	71.24	73.53	84.38	87.85	82.16	86.56	75.48	78.67	80.57	85.32
GIN	62.60	68.61	73.71	75.45	68.49	72.33	69.92	72.75	83.46	87.75	82.06	86.63	74.71	78.38	80.13	85.28
GAT	62.43	68.59	73.05	75.43	67.17	72.90	69.81	73.03	83.97	87.84	82.11	86.59	74.74	77.99	79.72	85.32
MPNN	69.59	71.09	78.25	79.52	71.17	74.01	72.91	74.92	84.79	**88.05**	83.58	87.12	78.32	80.01	81.19	85.55
CGC	67.73	70.69	77.99	79.43	70.04	73.93	71.27	74.47	84.66	87.97	83.67	86.97	76.16	78.99	81.08	85.25
Transformer	69.97	71.68	78.29	79.62	71.43	74.02	73.32	75.59	84.27	87.90	83.44	87.05	78.12	80.27	81.31	85.65
GEN	69.71	71.03	77.74	79.45	72.40	74.55	72.76	75.00	83.07	87.60	84.34	87.08	77.45	80.01	79.83	85.35
TRAVEL	**70.88**	**71.89**	**79.00**	**80.32**	**73.57**	**75.26**	**75.24**	**76.66**	**86.18**	87.94	**85.03**	**87.58**	**78.99**	**80.51**	**83.05**	**85.70**

A.4 Hyperparameter Robustness

In this experiment, we test the robustness of the models to different numbers of hidden units on the Dallas dataset. From Table 8, we observe that our TRAVEL model performs well robustly (and outperforms the baselines) across the entire range of hyperparameter values

A.5 Experimental Settings

All neural network models are implemented using PyTorch [20] version 1.10.0 and PyTorch Geometric [8] version 2.0.2. The models are trained using the cross-entropy loss and Adam optimizer. We train models for 300 epochs and use 16 as their hidden dimensions.

The experiments are run on a machine with Intel(R) Core(TM) i7-7700 CPU @ 3.60GHz and NVIDIA(R) GeForce(R) GTX 1060 graphics card (CUDA 10.2). Due to the large number of datasets

Table 6: Training time on the accident occurrence prediction task (in seconds).

Classifier	Houston	Charlotte	Dallas	Austin	LA	Atlanta	Seattle	Chicago
XGBoost	0.56	0.37	0.35	0.26	0.49	0.20	0.26	0.38
MLP	26.95	18.07	18.61	15.86	16.79	10.49	15.15	17.61
GCN	30.02	15.59	22.03	18.67	29.11	12.25	15.47	20.73
ChebNet	30.73	16.14	22.50	19.04	21.32	12.24	16.19	21.60
ARMANet	29.16	14.96	20.87	17.53	20.09	11.60	14.94	19.56
GraphSAGE	28.60	14.95	20.76	17.51	19.37	11.22	13.39	20.01
TAGCN	31.43	16.75	22.69	19.40	21.56	13.28	14.26	21.82
GIN	28.12	14.47	20.82	17.78	19.27	11.32	13.67	19.73
GAT	37.00	17.78	25.64	21.65	25.32	13.42	15.47	23.66
MPNN	49.72	21.40	33.48	33.25	33.98	18.96	24.21	30.74
CGC	31.83	15.75	23.29	19.80	25.92	11.70	13.48	21.29
Transformer	38.17	18.41	27.59	23.54	32.23	13.08	15.65	24.51
GEN	39.94	18.97	28.11	22.80	32.52	13.58	15.86	25.98
TRAVEL	47.57	21.03	32.80	26.37	39.05	15.60	17.87	29.25

Table 7: Training time on the accident severity prediction task (in seconds).

Classifier	Houston	Charlotte	Dallas	Austin	LA	Atlanta	Seattle	Chicago
XGBoost	3.10	1.60	1.81	1.25	3.00	0.80	1.21	1.97
MLP	16.34	11.27	12.01	10.95	13.70	8.74	9.86	11.91
GCN	20.36	13.68	14.40	13.47	15.99	10.27	11.64	14.33
ChebNet	20.26	14.05	14.97	13.01	16.57	10.50	11.74	14.37
ARMANet	19.30	12.83	13.75	11.94	15.06	9.46	11.00	12.95
GraphSAGE	18.30	12.70	13.66	11.75	15.28	9.37	10.99	12.98
TAGCN	21.89	14.18	15.34	13.06	17.34	10.63	12.22	14.72
GIN	18.33	12.76	13.29	11.64	14.72	9.41	10.59	13.53
GAT	24.47	16.74	18.09	15.01	21.87	12.06	13.06	16.56
MPNN	35.01	20.01	23.49	23.35	30.39	15.98	20.14	22.83
CGC	23.59	14.29	15.39	12.36	18.78	9.98	11.33	16.17
Transformer	28.65	16.78	18.73	15.18	22.95	12.07	13.38	18.09
GEN	28.45	16.85	18.73	15.12	24.25	11.70	13.97	18.52
TRAVEL	34.56	19.79	23.59	17.98	29.65	13.36	16.23	21.98

Table 8: Experiment results of the accident occurrence prediction task under different hidden dimension in terms of F1 score(%) and AUC(%) on the Dallas dataset.

	16 hidden units		32 hidden units		64 hidden units		128 hidden units	
Classifier	F1	AUC	F1	AUC	F1	AUC	F1	AUC
XGBoost	33.74	59.48	32.21	58.86	33.22	59.32	31.77	58.68
MLP	33.74	69.53	32.21	68.67	33.20	68.31	31.13	67.45
GCN	45.84	74.66	43.94	73.61	46.44	73.70	47.12	73.70
ChebNet	50.72	77.58	47.95	77.19	52.12	77.13	47.73	76.42
ARMANet	47.36	77.51	46.65	76.94	51.55	76.99	49.85	76.07
GraphSAGE	46.73	76.09	45.64	75.05	46.31	75.61	49.93	75.28
TAGCN	54.88	78.70	50.43	78.06	53.85	78.22	52.02	77.93
GIN	48.47	76.34	47.21	75.96	51.30	77.09	51.37	75.76
GAT	47.24	75.81	44.94	75.59	46.81	75.99	47.40	75.29
MPNN	55.39	81.37	54.85	81.64	57.79	81.78	60.05	81.70
CGC	52.29	80.08	51.83	80.42	51.40	79.77	55.40	80.79
Transformer	58.07	81.92	59.07	82.40	59.30	81.75	59.05	81.97
GEN	60.91	81.89	63.52	82.42	62.41	80.30	62.86	81.86
TRAVEL	**64.40**	**83.02**	**64.73**	**82.84**	**63.71**	**82.00**	**63.83**	**82.55**

(1000) as well as baselines, the experiments take close to 90 hours to run in total.

All GNN models have two graph convolutional layers, where layer-2 embeddings get information from nodes two hops away. We also add a fully connected layer after GNN layers to increase their expressive power. Moreover, L2 penalty and dropout layers with 0.5 dropout rates are added to all neural network models to reduce overfitting.