Graph Sequence Neural Network with an Attention Mechanism for Traffic Speed Prediction

ZHILONG LU, WEIFENG LV, ZHIPU XIE, BOWEN DU, GUIXI XIONG, and LEILEI SUN,
State Key Laboratory of Software Development Environment, Beihang University, China
HAIQUAN WANG, School of Software, Beihang University, China

Recent years have witnessed the emerging success of Graph Neural Networks (GNNs) for modeling graphical data. A GNN can model the spatial dependencies of nodes in a graph based on message passing through node aggregation. However, in many application scenarios, these spatial dependencies can change over time, and a basic GNN model cannot capture these changes. In this article, we propose a **G**raph **Se**quence neural network with an **At**tention mechanism (GSeqAtt) for processing graph sequences. More specifically, two attention mechanisms are combined: a horizontal mechanism and a vertical mechanism. GTransformer, which is a horizontal attention mechanism for handling time series, is used to capture the correlations between graphs in the input time sequence. The vertical attention mechanism, a Graph Network (GN) block structure with an attention mechanism (GNAtt), acts within the graph structure in each frame of the time series. Experiments show that our proposed model is able to handle information propagation for graph sequences accurately and efficiently. Moreover, results on real-world data from three road intersections show that our GSeqAtt outperforms state-of-the-art baselines on the traffic speed prediction task.

CCS Concepts: • **Information systems** → **Spatial-temporal systems;**

Additional Key Words and Phrases: Graph neural network, self-attention, traffic speed prediction

ACM Reference format:
Zhilong Lu, Weifeng Lv, Zhipu Xie, Bowen Du, Guixi Xiong, Leilei Sun, and Haiquan Wang. 2022. Graph Sequence Neural Network with an Attention Mechanism for Traffic Speed Prediction. *ACM Trans. Intell. Syst. Technol.* 13, 2, Article 20 (February 2022), 24 pages.
https://doi.org/10.1145/3470889

This work was supported by the National Key Technology Support Program of china under Grant No. 2019YFF0301302-2 and Guangxi Innovation-Driven Development Special Fund Project under Grant No. AA18118053, and the National Natural Science Foundation of China under Grant Nos. 51778033, 51822802, 51991395, 71901011, and U1811463, and the Beijing Municipal Science and Technology Project under Grant Z191100002519012.

Authors' addresses: Z. Lu, W. Lv, Z. Xie, B. Du, G. Xiong, and L. Sun, State Key Laboratory of Software Development Environment, Beihang University, Haidian District, No. 37 Xueyuan Road, Beijing, China; emails: {zhilong_lu, lwf, xiezhipu, dubowen, xiongguixi, leileisun}@buaa.edu.cn; H. Wang (corresponding author), School of Software, Beihang University, Haidian District, No. 37 Xueyuan Road, Beijing, China; email: whq@buaa.edu.cn.

2157-6904/2022/02-ART20 $15.00
https://doi.org/10.1145/3470889

1 INTRODUCTION

Traffic speed prediction is one of the most challenging tasks in **Intelligent Transportation Systems (ITSs)**. Accurate and reliable road speed prediction not only can benefit city management departments by providing tools for policymaking but also can provide individual travelers with sufficient information for making plans [1]. Thus, traffic speed prediction has been extensively studied in recent years. Many conventional shadow models, such as **Support Vector Machine (SVM)** models [2] and the **AutoRegressive Integrated Moving Average (ARIMA)** model [3], can handle time-series data but cannot capture the spatio-temporal relationships of road networks. However, with the development of deep learning, many methods have emerged that can capture temporal and spatial properties separately or even simultaneously. **Long Short-Term Memory (LSTM)** [4] shows superior performance in capturing temporal correlations related to traffic conditions. **Convolutional Neural Networks (CNNs)**, such as those presented in [5, 6], can capture dependencies in Euclidean space for modeling spatial relations. An end-to-end deep learning framework has been proposed that can be used to estimate the travel time between any two points between cities, in contrast to the traditional methods of estimating the travel time for each section and then summing the results, which will lead to error accumulation [7]. In recent years, graph-based deep learning methods have been gradually applied in many fields. In these methods, a transportation network is treated as a graph, on which traffic speed predictions are formulated, and **Graph Convolutional Networks (GCNs)** are then employed to model the non-Euclidean correlations in the road network, as done in [8, 9].

In reality, many types of data, such as social network data, transfer relationship data, and molecular structure data, cannot be simply modeled as tensors; instead, these kinds of data are often organized in the form of graphs [10]. Data that can be modeled as tensors (such as 1D text, 2D images, and 3D videos) are named Euclidean data, while data modeled as graphs are named non-Euclidean data. For information that can be naturally represented as a graph, the traditional approach is to transform the graph into a series of flat vectors. However, this approach eliminates the topological information that can be expressed by the graph, such as the edge information and the node context, which is difficult to express in flat vectors, and the consequent lack of information is likely to lead to a decline in the modeling effect. To fully retain the topological information among the nodes in the graph when processing knowledge in graph form, a more efficient way is to use a **Graph Neural Network (GNN)**. GNNs can be traced back to 2004, when Scarselli and Gori carried out a series of related studies [11]. GNNs are neural networks that can model graphs directly based on **Recursive Neural Networks (RNNs)** [12] but with a slightly different cycle. GNNs can be applied for various forms of machine learning based on graphs, including supervised learning, unsupervised learning, and reinforcement learning. Many scholars have proposed a variety of different ways to implement GNNs, such as GCNs [13], **Graph Attention networks (GATs)** [14], and **Graph Network (GN)** blocks [15], and GNNs based on various applications have also been studied. Meanwhile, the concept of "attention" has recently gained popularity in regard to the training of neural networks. Attention mechanisms allow models to learn alignments between different modalities, e.g., between image objects and agent actions in a dynamic control problem [16] or between the visual features of an image and its text description for the image caption generation task [17]. Deep neural networks with attention operators have shown remarkable capabilities in solving challenging tasks in various fields, such as natural language processing, computer vision, and network embedding [18]. Bahdanau et al. [19] proposed an attention mechanism for the Seq2Seq framework in the context of indexing past historical sequences for RNN-based language models. Vaswani et al. [20] proposed a sequence processing framework using only an attention mechanism, named Transformer. Compared with sequence processing models based on RNNs, Transformer has not only a better prediction effect but also better parallelism and a higher training efficiency. In particular, attention mechanisms have played an important role in many tasks related to traffic speed prediction. Unlike a traditional multi-head attention mechanism, which considers all attention heads equally, a GaAN uses a convolutional sub-network to control each attention head's importance [21]. A novel attention-based spatio-temporal graph convolution model named ASTGCN has also been proposed and successfully applied to forecast traffic flows [22]. Focusing on spatio-temporal factors, Zheng et al. [23] proposed a **Graph Multi-Attention Network (GMAN)** to predict traffic conditions at future time steps in various locations on a road network graph. Lin et al. [24] studied the impacts of irrelevant noise and error propagation on the long-term prediction of spatio-temporal phenomena and proposed a **Dynamic Switch-Attention Network (DSAN)** with a novel **Multi-Space Attention (MSA)** mechanism to explicitly measure the correlations between the inputs and outputs.

Among the various ways of realizing a GNN, there is only one that ensures the isomorphism of the graph while treating different elements differently, that is, a GAT. The difference is that in the process of constructing a GAT, the attribute information associated with each edge is not considered, nor are the hidden states of the edge connections represented in the process of information transmission. Therefore, when the information of a node is updated, information is propagated from the neighbouring nodes to the target node. Even in this case, if the traffic invariance of the graph is to be guaranteed, the weights cannot be arbitrarily set for different neighbouring nodes. This is where the attention mechanism employed in a GAT comes into play. The essential purpose of an attention mechanism is to determine the correlations between potentially relevant information and target elements by measuring the similarity between elements. In accordance with these correlations, the weights of the relevant information with respect to the target elements are calculated. When an attention mechanism is used in the encoding stage, a target element and its context are usually summarized in terms of these weights to map the feature vector associated with the target element to an implicit state that accounts for its context. This kind of attention mechanism is named a self-attention mechanism. A GAT adopts precisely such a self-attention mechanism and differentially summarizes the information of neighbouring nodes by means of the weights determined by the attention function to map the attributes of the nodes step by step to increasingly deeper hidden states. Therefore, we believe that on the basis of the attention mechanism used in GATs, we can realize an information aggregation algorithm that can act on GN blocks to summarize edge information in a different way, which can be used to update the attributes of corresponding nodes. On the other hand, in the time dimension, the RNN concept can still be used to process graph sequences. Most techniques for time-series processing based on deep learning are borrowed from the field of text processing. In text processing, RNNs have always been regarded as the standard class of models for processing text sequences. This is largely due to the flexible processing power of RNNs for variable-length sequences and their good performance in text processing. However, the shortcomings of RNNs due to their own structure, including the difficulty of capturing long-distance dependencies and the inability to train RNNs in parallel, have not yet been effectively overcome. In recent years, the academic community has begun to abandon RNNs in favor of using other forms of deep sequence networks to process text sequences, including **Temporal Convolutional Networks (TCNs)** and Transformers. In particular, the Transformer framework, based on only an attention mechanism, has been gradually replacing RNNs as the dominant approach in text sequence processing.

There are two challenges that need to be solved in this article; the first one is how to combine the attention mechanism with a GN block to capture inherent correlations between different nodes and edges in a graph, and the other one is how to capture long-distance dependencies for a graph sequence as input with an attention mechanism. Finally, we propose a graph sequence neural network with an attention mechanism, which is a two-phase framework named GSeqAtt, to learn representations of the spatio-temporal relations of graph-structured data. Specifically, GSeqAtt contains two types of attention mechanisms: a horizontal mechanism and a vertical mechanism. GTransformer, which is a horizontal attention mechanism for handling time series, is used to capture the correlations between graphs in the input time sequence. The vertical attention mechanism, a GN block structure with an attention mechanism (GNAtt), acts within the graph structure in each frame of the time series. As the basis for this mechanism, we combine the GN block structure with the GAT structure to construct a new neural network unit to capture vertical attention information within a GN block in a sequential GNN.

In summary, the major contributions of this article include the following:

- **A new GN block with an attention mechanism.** The basic GN block ignores edge importance during the updating of node information. We propose a new block named GNAtt that incorporates an attention mechanism by revising the update procedure to assign different levels of importance to related edges when updating nodes.
- **A graph model with Transformer.** We propose a new model named GTransformer that adopts the self-attention mechanism of Transformer to capture the relations between graphs in the input sequence.
- **Comprehensive empirical evaluation using real-world data.** We evaluate GSeqAtt by conducting a comprehensive evaluation on the traffic speed prediction task under traffic control using real-world datasets from three selected roads.

The rest of this article is organized as follows. Section 2 describes the related work. Section 3 presents the problem definition. Section 4 introduces our proposed methodology. Section 5 elaborates on our measurement results and provides some insights and discussion, followed by a conclusion in Section 6.

2 RELATED WORK

2.1 Traffic Speed Prediction

For urban travelers, the use of navigation systems that can predict the spread of congestion in advance can help avoid potentially congested roads, save travel time, and thus improve travel efficiency. Guo et al. [25] proposed a real-time path planning algorithm based on real-time **Travel Time Estimation (TTE)** to avoid congested roads, which can not only save travelers travel time but also effectively reduce the duration of traffic congestion. In early studies on road segment speed prediction [26], vehicle passage speeds at several consecutive moments were modeled as a time series and predicted using the **AutoRegressive Moving Average (ARMA)** model. Chen et al. [27] investigated the use of the **K-Nearest Neighbor (KNN)** algorithm in traffic speed prediction and achieved slightly better results than those of other state-of-the-art techniques. A different class of time-series models named **Structural Time-series Models (STMs)** (in their multivariate form) has also been introduced to develop a parsimonious and computationally simple multivariate short-term traffic condition forecasting algorithm in [28]. In recent years, traffic speed prediction methods based on deep learning have gradually become a popular research topic. Huang and Ran [29] proposed a deep neural network model for traffic speed prediction. The most prominent feature of this model is that the influence of extreme weather on traffic is considered in the input to the neural network. Khotanzad and Sadek [30] predicted highway segment velocity by using a **MultiLayer Perceptron (MLP)** and a **Fuzzy Neural Network (FNN)**, and experiments showed that the proposed method offered significantly improved results compared with a traditional time-series analysis algorithm. Wang et al. [31] proposed a deep learning method with an **error feedback Recurrent Convolutional Neural Network structure (eRCNN)** for continuous traffic speed prediction. A prediction model named a deep **Spatio-Temporal Residual Network**

(ST-ResNet) has also been designed for traffic speed prediction, in which the residual network structure is conducive to information transmission within the deep neural network [32]. Spatio-temporal traffic data can be converted into an image in which the traffic data are expressed in a 3D space with respect to space and time axes. As an alternative to CNNs, which lose important information by locally taking the highest activation values, Kim et al. [33] proposed a neural network with capsules in which max pooling is replaced by dynamic routing for traffic speed prediction. Liang et al. [34] added a multistage attention mechanism for the prediction of spatio-temporal time series and considered other external factors in the prediction model through a fusion module. Kim et al. [35] demonstrated that embedding information on the topology of the road network improved the process of learning traffic features and then applied RNNs to a graph of a vehicular road network to infer the interactions between adjacent road segments as well as the temporal dynamics. Although there are many methods of traffic speed prediction, emerging neural models can be applied to enhance the accuracy of forecasting.

2.2 Neural Networks Based on Graphs

Graphs are widely used data structures for describing associations among entities; for example, for the interpersonal relationships in a social network, a graph is usually used as the basic data structure. When we wish to use a neural network to mine useful patterns from the structural information represented by a graph, it is necessary to use a GNN. Zhou et al. [36] stated that a GNN is a deep learning method that can be used to process graph-structured data. GNNs can be traced back to 2004, when Scarselli and Gori carried out a series of related studies [11]. GNNs are neural networks that can model graphs directly based on RNNs [12] but with a slightly different cycle. GNNs can be applied for various forms of machine learning based on graphs, such as supervised learning, unsupervised learning, and reinforcement learning. Following its proposal, this method was quickly applied in practical application scenarios such as web page ranking [37]. Moreover, Scarselli et al. [38] performed a detailed analysis of the computational performance of the web page ranking algorithm cited above. Till today, there have been many studies on GNNs. Scarselli et al. proposed a GNN model for mapping a graph G and all its nodes N to an M-dimensional Euclidean space [39]. On the basis of previous work, Li et al. [40] proposed a gated graph sequence neural network by extending the cycle of the gate control unit, allowing the GNN to generate output in the form of a graph sequence. After decades of development, the stature of GNNs in the field has gradually changed. Especially in recent years, the development of GNNs has become increasingly mature, and many meaningful new studies have appeared. Many scholars have proposed a variety of different ways of implementing GNNs, such as GCNs [13], GATs [14], and GN blocks [15], and GNNs based on various applications have also been studied. Bruna et al. [41] proposed a graph convolution operation defined on the basis of the graph Fourier transform and its inverse transformation and introduced this operation into neural networks. Two methods of constructing graph convolutions were proposed: one based on spectral theory and another based on spatial relations. Gilmer et al. [42] unified GNNs and GCNs into **Message-Passing Neural Networks (MPNNs)** and tested other, more effective variants of this framework. GNNs have received extensive attention and have been applied in various fields, which may reflect the urgent need in the industry for a means of effectively processing graph-structured data. In particular, Li et al. [9] and Cui et al. [43] both utilized GNNs for traffic speed prediction.

2.3 Attention Mechanisms

The development of attention mechanisms was originally motivated by the need for a recurrent attention model for computer vision to reduce the amount of computation performed by deep neural networks, inspired by the fact that the human visual system pays attention to only a portion of the important information in a scene rather than the whole scene [16]. Bahdanau et al. [19] proposed an attention mechanism for the Seq2Seq framework in the context of indexing past historical sequences for RNN-based language models to solve the performance bottleneck caused by information propagation between the encoder and decoder through only fixed-length state coding. Two types of attention mechanisms can be identified: global attention mechanisms focusing on the whole input sequence and local attention mechanisms focusing on only a subset of the input sequence [44]. It has been proven that with the incorporation of an attention mechanism, the prediction effect of a RNN model is significantly improved. Vaswani et al. [20] proposed a sequence processing framework using only an attention mechanism, named Transformer. Compared with sequence processing models based on RNNs, Transformer has not only a better prediction effect but also better parallelism and a higher training efficiency. An attention mechanism enables a deep neural network to gather the context to make predictions by focusing computational efforts on a certain subset of model inputs or layer-wise features [45]. Attention mechanisms are widely adopted in the machine learning field, and many variants have been proposed for many different applications. Wang et al. [46] proposed a **Heterogeneous graph Attention Network (HAN)** based on a hierarchical attention mechanism. When dealing with the properties of nodes, the problem arises that different types of nodes have different characteristic vector lengths, and even if the length is the same, the individual dimensions may have different meanings, making it unreasonable to combine the vectors for different types of nodes into a single characteristic matrix. The hierarchical attention mechanism of a HAN can overcome this issue. Velickovic et al. [14] used a self-attention mechanism in place of the convolution mechanism when implementing a GNN. The hidden state for each node was updated by using the weighted average of the neighboring nodes, where the weights used were determined by the degrees of similarity between the node being processed and its neighboring nodes.

3 PROBLEM DEFINITION

In this section, we first introduce the notions of a graph in the context of processing by a GNN, a road network, and a linkage graph, and we then define the target problem.

3.1 Preliminaries

We first define a graph in the context of processing by a GNN, a road network, and a linkage graph.

Definition 1 (Graph to Be Processed by a GNN). A normal graph is composed of nodes and edges connecting the nodes. However, a graph that is processed by a GNN is defined with some additional properties. Specifically, such a graph is defined as $G = (V, E, V^{attr}, E^{attr}, U^{attr})$, where V is the set of nodes and E is the set of edges. V represents all of the nodes (or vertices) in the graph, while $E = (e_k, s_k, r_k)$ contains all of the edges. v_j and e_k are small integers sorted in ascending order. The sender and receiver of an edge are both nodes in V and are represented by s_k and r_k, respectively. V^{attr} contains the attributes of all nodes. The ith row of V^{attr} is the attribute vector of the ith node. Similarly, E^{attr} is a matrix that contains the attributes of all edges, and the ith row of E^{attr} is the attribute vector of the ith edge. Finally, U^{attr} represents the attribute vector of the whole graph.

Definition 2 (Road Network). We model a road network as a directed graph $G = (V, E)$. Each vertex $v \in V$ denotes an intersection or a segmentation point of a road, and each edge $e \in E$ represents a road segment that links contiguous intersections. A directional edge $e \in E$ from intersection v_i to intersection v_j is established when there is a road segment between v_i and v_j. $V = \{v_1, v_2, \ldots, v_N\}$, where N is the number of road intersections. $E = \{e_1, e_2, \ldots, e_M\}$, where M is the number of road segments. Figure 1(a) illustrates the graph structure of a road network. There

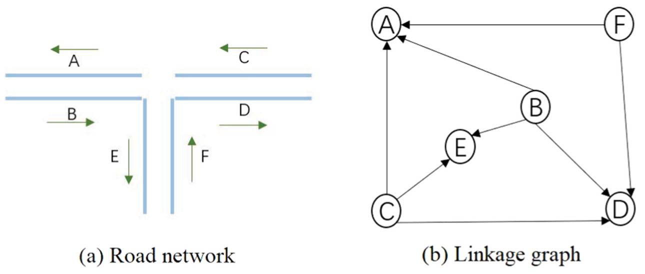


Fig. 1. Differences between two modeling structures.

is a road intersection selected with six segments named A to F. Segment A and C are in the same direction from right to left, while B and D are from left to right. The E is from top to bottom, while F has the opposite direction.

Definition 3 (Linkage Graph). For handling a simple graph such that the useful information contained in the vertices and edges can be combined, a linkage-graph-structured network [47] is adopted, which is different from a normal road network. $G^* = (V^*, E^*)$ denotes the new linkage graph. Each vertex $v^* \in V^*$ represents a road segment, and each edge $e^* \in E^*$ represents a direct link between adjacent segments. $V^* = \{v_1^*, v_2^*, \ldots, v_M^*\}$, where M is the number of road segments. $E^* = \{e_1^*, e_2^*, \ldots, e_L^*\}$, where L is the number of linkages. A directional edge e_i^* from segment v_i^* to segment v_j^* will be established if there is an approachable path from v_i^* to v_j^*. Compared to G as defined above, each node in G^* represents a road segment $e \in E$, and each edge represents a reachable direct link between two segments $e \in E$. From the road network shown in Figure 1(a), we can find that segment B, C, and F can approach segment A and D directly. Segment E can be approached by B and C directly. In the linkage graph, vertex represents different segments, while edges denote the link between these segments. After the transformation, we can get Figure 1(b), which illustrates the graph structure of the linkage network in Figure 1(a).

3.2 Problem Statement

Given data on the road segments from the past time intervals from t_{j-k} to t_{j-1}, the problem is to predict the road traffic speeds on road segments v_i^* during peak hours from t_j to t_{j+l} for several connected road segments simultaneously. We wish to predict $Y = \{S_{t_r}^{v_i^*} \mid r = j, \ldots, j+l\}$ given historical data $\{S_{t_r}^{v_i^*} \mid r = j-k, \ldots, j-1\}$, where $i \in \{1, 2, \ldots, M\}$, k represents the smallest time interval, and $S_{t_j}^{v_i^*}$ denotes the traffic speed on road segment v_i^* at t_j, while $\hat{S}_{t_j}^{v_i^*}$ denotes the predicted value of this speed. The problem can be formalized as follows:

$$\left\{S_{t_r}^{v_i^*} \mid r = j, \ldots, j+l\right\} = \mathcal{F}\left\{S_{t_r}^{v_i^*} \mid r = j-k, \ldots, j-1\right\}. \tag{1}$$

4 PROPOSED METHODOLOGY

4.1 Overall Framework

In this section, we present a detailed description of our proposed model, a graph sequence neural network architecture based on a mixed attention mechanism, named GSeqAtt. This model has two main components: GNAtt and GTransformer. First, based on the attention mechanism adopted in the GN framework and the GAT model, a GNAtt instance is constructed as a GN block with an

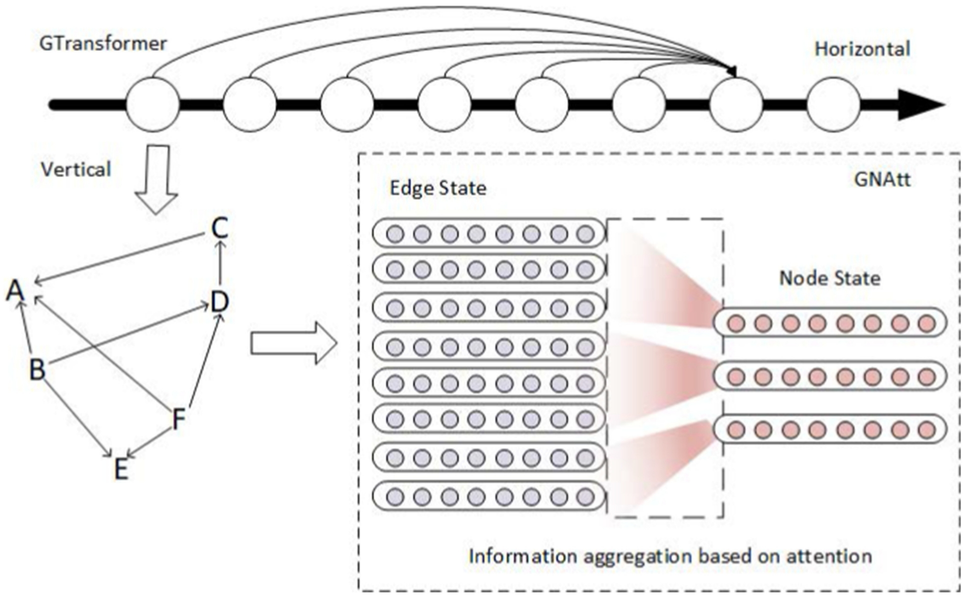


Fig. 2. Structure diagram of the proposed neural network for processing graph sequences based on an attention mechanism.

attention mechanism. This structure combines the advantages of GATs and GN blocks; specifically, it can not only process graph-structured data with edge information but also realize differential information transmission during the process of information aggregation. Then, a graph sequence processing structure named GTransformer is constructed based on an attention model. This structure can realize information propagation in the time dimension in the graph sequence neural network. Parallel training is also supported to improve the training efficiency of the neural network model. The two attention mechanisms can be used simultaneously in the same neural network model by means of alternating information transmission. Finally, the neural network structure GSeqAtt, in which both attention mechanisms act on the input graph sequence simultaneously, is formed. The overall framework of the proposed model is shown in Figure 2.

4.2 GN Block for Graph to Graph

A new GN framework was presented that defines a class of functions for relational reasoning over graph-structured representations [15]. The main unit of computation in the GN framework is the GN block, which takes a graph as input and returns a graph. Moreover, the GN block can be constructed for various complex architectures when combined with other models. A GN block, as defined in Equation (2), is a "graph-to-graph" module, which means that it takes a graph as input, denoted as $G = (V, E, V^{attr}, E^{attr}, U^{attr})$, and returns a graph denoted as $G' = (V', E', V^{attr'}, E^{attr'}, U^{attr'})$ as output:

$$G' = \mathbf{GN}(G). \tag{2}$$

Graphs G' and G are isomorphic, which means that $V' = V$ and $E' = E$. The computation in a GN block is described in Algorithm 1. In the GN block, $V^{attr'}$ is considered to be associated with V^{attr}, as well as E^{attr} and U^{attr}. These additional attributes are merged into a new matrix, denoted as $\widetilde{E}^{attr}$.

ALGORITHM 1: Computation in a GN block

```
Input:
    $G = (V, E, V^{attr}, E^{attr}, U^{attr})$
Output:
    $G' = (V', E', V^{attr'}, E^{attr'}, U^{attr'})$
1: for i in number of E do
2:     $\widetilde{E}_i^{attr} = [E_i^{attr}, V_r^{attr}, V_s^{attr}, U^{attr}]$
3: end for
4: $E^{attr'} = tanh(W_E * \widetilde{E}^{attr} + b_E)$
5: for i in number of V do
6:     $\widetilde{V}_i^{attr} = [agg_{e\to v}^i(E^{attr'}), V_i^{attr}, U^{attr}]$, where $agg_{e\to v}^i(E^{attr'}) = \sum_{j\in\{j|E_j.r_k=V_i\}} E_j^{attr'}$
7: end for
8: $V^{attr'} = tanh(W_V * \widetilde{V}^{attr} + b_V)$
9: $\widetilde{U}^{attr} = [\sum E^{attr'}, \sum V^{attr'}, U^{attr}]$
10: $U^{attr'} = tanh(W_U * \widetilde{U}^{attr} + b_U)$
```

4.3 GN Block with an Attention Mechanism

To realize GNAtt, we need to modify an aggregation function in the GN block, specifically the aggregation function ρ^{e2v} that applied to the incoming edges e of a node v, from the original average aggregation function $agg_{e\to v}^i$ as shown in Algorithm 1 to differential weighted aggregation. We adopt the GAT concept and use an attention mechanism to calculate the attention coefficient between the target object and each other object in the context. The original ρ_{ori}^{e2v} in the GN block is formulated as shown in Equation (3):

$$\rho_{ori}^{e2v} = \frac{1}{|inEdge(v)|} \sum_{e_{w,v}\in inEdge(v)} h_{e_{w,v}}^l, \tag{3}$$

where $inEdge(v)$ represents the set of all incoming edges of node v, i.e., all edges with node v as the receiver node. $e_{w,v}$ denotes the edge with the sender node w and the receiver node v. $|inEdge(v)|$ represents the number of incoming edges of node v, and $h_{e_{w,v}}^l$ denotes the hidden state of $e_{w,v}$ after the layer l update of the GN block. Actually, the updating of node attributes follows the updating of edge attributes in the GN block. And $h_{e_{w,v}}^l$ in the GN block is calculated as shown in Equation (4):

$$h_{e_{w,v}}^l = \phi^e(e_{w,v}, w, v, U^{attr}). \tag{4}$$

The ϕ^e is mapped across all edges to compute per-edge updates. In this article, the ϕ^e is factored into the scalar pairwise-interaction function, which returns the unnormalized attention term. By analogy to the GAT model, ρ_{att}^{e2v} with an attention mechanism can be expressed as shown in Equation (5):

$$\rho_{att}^{e2v} = \sum_{e_{w,v}\in inEdge(v)} \alpha_{(v,e_{w,v})} * h_{e_{w,v}}^l. \tag{5}$$

We find that the aggregation function before the addition of the attention mechanism is equivalent to giving the same weight of $\frac{1}{|inEdge(v)|}$ to each edge, while after the attention mechanism is introduced, each edge has a different weight of $\alpha_{(v,e_{w,v})}$. Thus, a reasonable design of the weights $\alpha_{(v,e_{w,v})}$ is of key importance for implementing ρ_{att}^{e2v}.

Information aggregation in a GAT mainly takes place between nodes and their neighbors; therefore, only the attention coefficients between neighboring nodes need to be calculated. By contrast, the updating of nodes in a GN block mainly involves the aggregation of incoming information, so it is necessary to design a method of calculating the attention coefficients between edges and nodes. Such a method will still roughly consist of the same steps in GAT but with slightly different objects. First, during the dense operation in GNAtt, the query vector is the attribute vector or hidden state of a node, while the value vector is the attribute vector or hidden state of an edge. The lengths of these two vectors are often different; therefore, it is impossible to use the same matrix to linearly transform them. Instead, it is necessary to use two different matrices for the linear transformation of the query and value vectors. The query vector is formulated as h_v^{l-1}, representing the hidden state of node v; let the length of this vector be FV. The value vector is $h_{e_{w,v}}^l$ representing the hidden state of an edge pointing from node w to node v, and the length of this vector is FE. The two matrices used for linear transformation are $W_V \in \mathbb{R}^{FV' \times FV}$ and $W_E \in \mathbb{R}^{FE' \times FE}$, and the results after linear transformation are $W_V h_v$ and $W_E h_{e_{w,v}}$, respectively. Subsequently, the concatenation operation is performed; that is, the two matrices are spliced together, followed by the use of the Dense(1) operation to convert the spliced vector into a vector of length 1, to which an activation function is applied. Then, the softmax function is called to obtain the final attention coefficient. The whole process can be formulated as shown in Equation (6):

$$\alpha_{(v,e_{w,v})} = \frac{\exp\left(LeakyReLU\left(DS1\left(W_v h_v^{l-1}; W_E h_{e_{w,v}}^l\right)\right)\right)}{\sum_{e_{w,k} \in inEdge(v)} \exp\left(LeakyReLU\left(DS1\left(W_v h_v^{l-1}; W_E h_{e_{k,v}}^l\right)\right)\right)}. \tag{6}$$

Here, $LeakyReLU(*)$ denotes the chosen activation function, and $DS1(*)$ denotes the Dense(1) transformation. Considering that $h_{e_{w,v}}^l$ is given by a linear transformation on $[h_{e_{vw}}^{l-1}; h_v^{l-1}; h_w^{l-1}; h_{\mathcal{G}}^{l-1}]$, the linear transformation on $[W_V h_v^{l-1}; W_E h_{e_{w,v}}^l]$ can be simplified as $h_{e_{w,v}}^l$. The simplified $\alpha_{(v,e_{w,v})}$ can be formulated as shown in Equation (7):

$$\alpha_{(v,e_{w,v})} = \frac{\exp\left(LeakyReLU\left(DS1\left(h_{e_{w,v}}^l\right)\right)\right)}{\sum_{e_{w,k} \in inEdge(v)} \exp\left(LeakyReLU\left(DS1\left(h_{e_{k,v}}^l\right)\right)\right)}. \tag{7}$$

Once $\alpha_{(v,e_{w,v})}$ has been computed and can be introduced into ρ_{att}^{e2v}, the final form of the information aggregation function based on an attention mechanism is obtained. Then, by combining this new aggregation function with the three update functions designed for the GN block, namely, ϕ^e, ϕ^v, and ϕ^g, and the other two aggregation functions, namely, ρ^{e2g} and ρ^{v2g}, the basic unit of GNAtt is obtained to replace the originally designed GN block unit. It should be noted that the GN block involves three aggregation functions in total. Theoretically, the other two aggregation functions could also be completely transformed into a form consistent with differential aggregation in accordance with the attention mechanism; this is an optional setting for GNAtt implementation. In this article, however, considering the enormous demand for computing resources introduced by the attention mechanism, the chosen solution is to modify only ρ^{e2v} to the attention-based differential aggregation form ρ_{att}^{e2v}, while the remaining two aggregation functions, ρ^{e2g} and ρ^{v2g}, retain their original form of simple averaging.

4.4 Graph Model with Transformer

A vertical attention mechanism can address only the propagation of information within a graph. GSeqAtt also requires the implementation of a temporal self-attention mechanism to handle a sequence of graphs, which is achieved by borrowing ideas from the Transformer model; hence, the proposed model is named GTransformer.

Consider a general sequence of input graphs $\mathcal{GSEQ\_IN} = [\mathcal{G}_{in}^1, \mathcal{G}_{in}^2, \ldots, \mathcal{G}_{in}^T]$, where $[\mathcal{G}_{in}^1, \mathcal{G}_{in}^2, \ldots, \mathcal{G}_{in}^T]$ is a set of isomorphic graphs with T time periods whose topological structure can be expressed as $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. The purpose of GTransformer is to handle $\mathcal{GSEQ\_IN}$

ALGORITHM 2: Single-layer Transformer function

```
Input:
    A sequence $[x_{ele}^1, x_{ele}^2, \dots, x_{ele}^T]$ in which the characteristics of a target element change over
    time.
Output:
    $[x_{ele}^1, x_{ele}^2, \dots, x_{ele}^T]$ re-encoded as $[EMB\_hid_{ele}^1, EMB\_hid_{ele}^1, \dots, EMB\_hid_{ele}^1]$
1: Apply a linear transformation to $x_{ele}$ and change the dimensions of $EMB_{x_{ele}}$, which is $x_{ele}$
   after the transformation, to $FS * HS$
2: Add information encoding to $EMB_{x_{ele}}$: $EMB_{raw} = EMB_{x_{ele}} + EMB_{TimeInfo}$
3: Use $Q\_MH = query\_mh(EMB_{raw})$ to obtain $Q\_MH$
4: Use $K\_MH = key\_mh(EMB_{raw})$ to obtain $K\_MH$
5: Use $V\_MH = value\_mh(EMB_{raw})$ to obtain $V\_MH$
6: Calculate multi-head attention: $EMB_{MH\_ATT} = MH\_ATT(Q\_MH, K\_MH, V\_MH)$
7: Construct residual connections and layer normalization: $EMB_{LN} = LayerNorm(EMB_{raw} +
   EMB_{MH\_ATT})$
8: Add a feedforward network module: $EMB_{FF} = FeedForward(EMB_{LN})$
9: Calculate the output encoding: $EMB\_hid = LayerNorm(EMB_{FF} + EMB_{LN})$
```

and output a new graph sequence $\mathcal{GSEQ\_OUT} = [\mathcal{G}_{out}^1, \mathcal{G}_{out}^2, \dots, \mathcal{G}_{out}^T]$. The graphs in the output sequence are isomorphic to the graphs in the input sequence and can also be represented by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$. We refer to each edge of a graph, each node, and the entire graph itself as elements of the graph, formulated as $\mathcal{G}\_ele$, where $\mathcal{G}\_ele = \mathcal{V} \cup \mathcal{E} \cup \{G\}$. For each $ele \in \mathcal{G}\_ele$, there is a corresponding property x_{ele}^t related to ele in $\mathcal{G}_{in}^t$. We can obtain a sequence $[x_{ele}^1, x_{ele}^2, \dots, x_{ele}^T]$ by traversing all of the graphs in $\mathcal{GSEQ\_IN}$. After the Transformer model is used to process the sequence, we obtain the re-encoding $[EMB\_hid_{ele}^1, EMB\_hid_{ele}^1, \dots, EMB\_hid_{ele}^1]$, which considers the context of the time series. The property vectors of all elements during all time periods can be updated by traversing this re-encoding vector. Finally, the result is reintegrated into the form of a graph sequence, and $\mathcal{GSEQ\_OUT}$ is generated. This temporal self-attention mechanism for graph sequences is named GTransformer. We first need to package one single-layer Transformer function, as shown in Algorithm 2, where $LayerNorm(*)$ is a method of normalization within a layer and is a common regularization method for avoiding over-fitting, $EMB_{raw} + EMB_{MH\_ATT}$ and $EMB_{FF} + EMB_{LN}$ are residual link operations that are used to prevent the original input information from being lost too quickly in the deep network, and $FeedForward(*)$ refers to the general MLP structure. The output of the algorithm, EMB_hid, is the hidden state of $[x_{ele}^1, x_{ele}^2, \dots, x_{ele}^T]$ re-encoded by the single-layer Transformer. The input and output of Algorithm 2 have the same characteristic dimensions, so this algorithm can be directly stacked multiple times to form a multi-tier Transformer module.

Based on the above definition of the Transformer function, we can present the algorithm for GTransformer, as shown in Algorithm 3. The input and output of this algorithm are graph sequences, denoted by $\mathcal{GSEQ\_IN}$ and $\mathcal{GSEQ\_OUT}$, respectively. We can re-encode any elements $ele \in \mathcal{G}\_ele$ by means of the Transformer function and reformat the result as an entire sequence of graphs to utilize GTransformer. In practice, however, this element-by-element approach may place considerable strain on the available computational resources. Therefore, to simplify the calculation of the algorithm, the attributes of nodes, edges, and the whole graph are usually divided into three separated groups. In a real algorithm implementation, the Transformer function could also be applied only to certain elements, as required by the problem scenario; for example, the Transformer function could be implemented only on time series of node attributes

ALGORITHM 3: GTransformer function

```
Input:
    Graph sequence $\mathcal{GSEQ\_IN}$, where the graphs have the same topology but the attributes of
    the elements change over time; the nodes and edges are denoted by $\mathcal{V}$ and $\mathcal{E}$
Output:
    Graph sequence $\mathcal{GSEQ\_OUT}$ after the properties have been updated
 1: Initialize the sequence matrix $Seq_{\mathcal{V}}$ corresponding to the nodes as a null matrix
 2: for $v$ in $\mathcal{V}$ do
 3:     Construct the time series $Seq_v$ corresponding to $v$
 4:     Add $Seq_v$ to $Seq_{\mathcal{V}}$
 5: end for
 6: $Seq\_hid_{\mathcal{V}} = Transformer(Seq_{\mathcal{V}})$
 7: Initialize the sequence matrix $Seq_{\mathcal{E}}$ corresponding to the edges as a null matrix
 8: for $e$ in $\mathcal{E}$ do
 9:     Construct the time series $Seq_e$ corresponding to $e$
10:     Add $Seq_e$ to $Seq_{\mathcal{E}}$
11: end for
12: $Seq\_hid_{\mathcal{E}} = Transformer(Seq_{\mathcal{E}})$
13: Construct the time series $Seq_\mu$ of global graphs corresponding to $\mu$
14: $Seq\_hid_\mu = Transformer(Seq_\mu)$
15: $\{Seq\_hid_{\mathcal{V}}, Seq\_hid_{\mathcal{E}}, Seq\_hid_\mu\}$ is reorganized as a sequence of graphs named $\mathcal{GSEQ\_OUT}$
```

if only the node attributes are of interest. In this respect, the GTransformer model can be flexibly controlled in accordance with the application requirements.

4.5 Mixed Attention

We have introduced how two kinds of information transmission can be constructed for a graph sequence based on attention mechanisms. Vertical information transmission is implemented based on GNAtt, whose scope of action is limited to the graph corresponding to an individual frame of the time sequence. Horizontal information transmission is implemented based on GTransformer, by specifically using each node of the graph, each edge, and the attributes of the entire graph to constitute time series and using the Transformer mechanism to process each time series. In this article, our objective is to combine these two attention mechanisms to form a mixed attention mechanism with vertical and horizontal properties that can be used to process a sequence of graphs, thus forming the final framework for GSeqAtt. As a **Non-Local Neural Network (NLNN)** mechanism, the nature of the self-attention mechanism is to map the state vector h^l_{ele} of an element ele to a re-encoding h^{l+1}_{ele} that considers its context information, formulated as $h^{l+1}_{ele} = f(h^l_{ele}, \{h^l_c | c \in C\})$, where l denotes the layer number of the network and the context C of an element is usually a collection of other elements associated with that element. The state vector of a node v at time t, denoted by $h_v^{(l,t)}$, is related to two types of contextual information. One is the graph structure, i.e., the spatial context formed by the neighboring nodes of node v and its adjacent edges, denoted by C_S, and the other is the temporal context formed by the state vector of the node at other times, denoted by C_T. In our mixed attention mechanism for graph sequences, we need to map $h_v^{(l,t)}$ to $h_v^{(l+1,t)}$ while considering C_S and C_T simultaneously; this mapping problem can be formulated as $h_v^{l+1} = f(h_v^l, \{h_c^l | c \in C_S \cup C_T\})$. Therefore, the key to realizing the mixed attention mechanism is to design a reasonable $f(*)$.

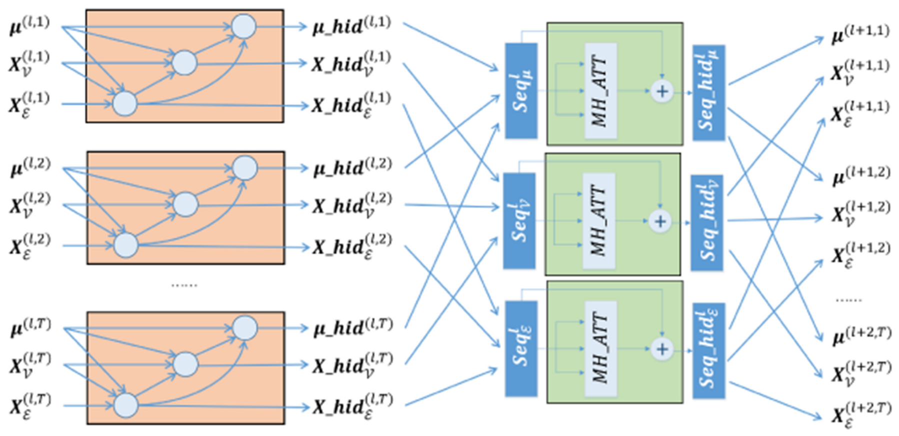

$\mu^{(l,1)}$
$X_{\mathcal{V}}^{(l,1)}$
$X_{\mathcal{E}}^{(l,1)}$
$\mu\_hid^{(l,1)}$
$X\_hid_{\mathcal{V}}^{(l,1)}$
$X\_hid_{\mathcal{E}}^{(l,1)}$
$\mu^{(l,2)}$
$X_{\mathcal{V}}^{(l,2)}$
$X_{\mathcal{E}}^{(l,2)}$
$\mu\_hid^{(l,2)}$
$X\_hid_{\mathcal{V}}^{(l,2)}$
$X\_hid_{\mathcal{E}}^{(l,2)}$
......
$\mu^{(l,T)}$
$X_{\mathcal{V}}^{(l,T)}$
$X_{\mathcal{E}}^{(l,T)}$
$\mu\_hid^{(l,T)}$
$X\_hid_{\mathcal{V}}^{(l,T)}$
$X\_hid_{\mathcal{E}}^{(l,T)}$
Seq_{μ}^{l}
$Seq_{\mathcal{V}}^{l}$
$Seq_{\mathcal{E}}^{l}$
MH_ATT
MH_ATT
MH_ATT
$Seq\_hid_{\mu}^{l}$
$Seq\_hid_{\mathcal{V}}^{l}$
$Seq\_hid_{\mathcal{E}}^{l}$
$\mu^{(l+1,1)}$
$X_{\mathcal{V}}^{(l+1,1)}$
$X_{\mathcal{E}}^{(l+1,1)}$
$\mu^{(l+1,2)}$
$X_{\mathcal{V}}^{(l+1,2)}$
$X_{\mathcal{E}}^{(l+1,2)}$
......
$\mu^{(l+2,T)}$
$X_{\mathcal{V}}^{(l+2,T)}$
$X_{\mathcal{E}}^{(l+2,T)}$

Fig. 3. Mixed attention mechanism in GSeqAtt.

We design the mixed attention mechanism as shown in Figure 3 based on the above considerations. This mixed attention mechanism works not only on the nodes but also on the edges and the entire graph. The input graph sequence corresponding to the mixed attention mechanism in layer l is denoted by $\mathcal{GSEQ}^{l} = [\mathcal{G}^{(l,1)}, \mathcal{G}^{(l,2)}, \ldots, \mathcal{G}^{(l,T)}]$, where $\mathcal{G}^{(l,t)}$ represents the graph state corresponding to time t and $\mathcal{G}^{(l,t)} = \{\mathcal{V}, \mathcal{E}, X_{\mathcal{V}}^{(l,t)}, X_{\mathcal{E}}^{(l,t)}, \mu^{(l,t)}\}$. Each $\mathcal{G}^{(l,t)}$ has the same topology, denoted by $\{\mathcal{V}, \mathcal{E}\}$. $X_{\mathcal{V}}^{(l,t)}$ represents the matrix of the state vectors of all nodes, $X_{\mathcal{E}}^{(l,t)}$ represents the matrix of the state vectors of all edges, and $\mu^{(l,t)}$ represents the matrix of the state vectors of the whole graph $\mathcal{G}^{(l,t)}$. After the mixed attention mechanism, the whole graph sequence is mapped to a new graph sequence $\mathcal{GSEQ}^{l+1} = [\mathcal{G}^{(l+1,1)}, \mathcal{G}^{(l+1,2)}, \ldots, \mathcal{G}^{(l+1,T)}]$, where $\mathcal{G}^{(l+1,t)} = \{\mathcal{V}, \mathcal{E}, X_{\mathcal{V}}^{(l+1,t)}, X_{\mathcal{E}}^{(l+1,t)}, \mu^{(l+1,t)}\}$. The new graph sequence has the same graph structure as the previous graph sequence, while the state vector for each element has changed. The whole process can be divided into two parts. First, $\mathcal{GSEQ}^{l}$ is mapped to $\mathcal{GSEQ\_HID}^{l} = [\mathcal{G\_HID}^{(l,1)}, \mathcal{G\_HID}^{(l,2)}, \ldots, \mathcal{G\_HID}^{(l,T)}]$ by GNAtt. Then, $\mathcal{GSEQ\_HID}^{l}$ is mapped to $\mathcal{GSEQ\_HID}^{l+1}$ by GTransformer. Taking the transformation process from $X_{\mathcal{V}}^{(l,t)}$ to $X_{\mathcal{V}}^{(l+1,t)}$ as an example, $X_{\mathcal{V}}^{(l,t)}$ is first mapped to the hidden state $X\_hid_{\mathcal{V}}^{(l,t)}$ by GNAtt, where the hidden state of each node contains its spatial context information in $X\_hid_{\mathcal{V}}^{(l,t)}$. After that, all of the $\{X\_hid_{\mathcal{V}}^{(l,t)}\}$ are reintegrated in chronological order to form a sequence matrix $Seq_{\mathcal{V}}^{l}$ representing the changes in the node state vectors over time. Then, $Seq_{\mathcal{V}}^{l}$ is mapped to a re-encoding $Seq\_hid_{\mathcal{V}}^{l}$ that contains the temporal context information obtained by GTransformer. Finally, all of the sequence data are recombined to form a re-encoding organized in accordance with the graph sequence, where $X_{\mathcal{V}}^{(l+1,t)}$ is the re-encoding corresponding to $X_{\mathcal{V}}^{(l,t)}$ and $X_{\mathcal{V}}^{(l+1,t)}$ simultaneously contains the spatial and temporal information for every node. A common feature of both GNAtt, which maps $\mathcal{GSEQ}^{l}$ to $\mathcal{GSEQ\_HID}^{l}$, and GTransformer, which maps $\mathcal{GSEQ\_HID}^{l}$ to $\mathcal{GSEQ}^{l+1}$, is that they leave the topology among the elements unchanged while applying attention mechanisms to generate re-encodings of the elements.

In the experiments presented in the next section, we investigate different iteration schemes to explore the best practices for GSeqAtt. Although similar to those for Transformer, the training process and reasoning process for GSeqAtt are slightly different. In the training phase, because the sequence mask operation is adopted, prediction can be carried out for a sequence composed of multiple graphs simultaneously. However, care is necessary when using a sequence mask. Specifically, any information on one predicted graph concerning the current time and future period cannot be used when predicting another graph; otherwise, information leakage will lead to large errors. In the reasoning stage, GSeqAtt can make only single-step predictions. To realize multi-step graph sequence prediction, the trained model needs to be called several times, and each time will generate a graph representing the traffic state of the road network in the next time slice in the future. Although we can theoretically achieve prediction for a graph sequence of infinite length by running the model many times, the accuracy of the model predictions will gradually decrease with an increasing amount of error propagation.

4.6 Objective Function and Optimization

The objective function is used to define the loss between predicted values and true values. In this model, we choose the **Mean Square Error (MSE)** as the loss function:

$$MSE = \frac{1}{T}\sum_{i=1}^{T} \left|S_t - \tilde{S}_t\right|^2, \tag{8}$$

where $\mathcal{L}$ is defined as follows:

$$\mathcal{L} = \frac{1}{M}\frac{1}{T}\sum_{i=1}^{M}\sum_{j=1}^{T} \left\|\hat{S}_{t_j}^{v_i^*} - S_{t_j}^{v_i^*}\right\|^2, \tag{9}$$

where T is the time span to be predicted, and M is the road segments for training and testing. Furthermore, $\hat{S}_{t_j}^{v_i^*}$ is the predicted road speed of a specific road segment v_i^* at t_j, while $S_{t_j}^{v_i^*}$ is the real road segment. We also select the Adam optimizer as the optimization algorithm in the training process.

5 EXPERIMENTS AND ANALYSIS

In this section, we present a comprehensive experimental study conducted on a real public dataset, the Q-Traffic dataset.[1] We describe the dataset, the compared methods, the implementation details, and the evaluation metrics. We also discuss the results of different models.

5.1 Dataset

The Q-Traffic dataset consists of three sub-datasets: a query sub-dataset, a traffic speed sub-dataset, and a road network sub-dataset. In this article, we use only the traffic speed and road network sub-datasets.

5.1.1 Traffic Speed Sub-dataset. Table 1 shows the statistics of the road segments. They are all located on the 6th Ring Road of Beijing, which is the most crowded traffic area in Beijing. The traffic speed on each road segment was recorded every 15 minutes.

5.1.2 Road Network Sub-dataset. Due to the spatio-temporal dependencies of traffic data, the topology of the road network is useful information for predicting traffic. Table 2 shows the fields of the road network sub-dataset.

[1]https://aistudio.baidu.com/aistudio/datasetDetail/76.

Table 1. Statistics of the Traffic Speed Sub-dataset

Item	Description
Road segments	15,073
Total length	738.91 km
Interval	15 minutes
Time	April 1, 2017–May 31, 2017
Total records	265,967,808
Long/lat bounding box	(116.10, 39.69, 116.71, 40.18)

Table 2. Examples of the Geographical Attributes of Each Road Segment

Field	Type	Description
link_id	Char(13)	road segment ID
width	Char(3)	width, 15: <=3.0 m; 30: (3.0 m, 5.0 m); 55: (5.5 m, 13 m); 130: >13 m
direction	Char(1)	direction, 0: unknown, default two-way; 1: two-way; 2: one-way, from start node to end node; 3: one-way, from end node to start node
snodeid	Char(13)	start node ID
enodeid	Char(13)	end node ID
length	Char(8)	length (km)
speedclass	Char(1)	speed limit (km/h), 1: >130; 2: (100, 130); 3: (90, 100); 4: (70, 90); 5: (50, 70); 6: (30, 50); 7: (11, 30); 8: <11
lanenum	Char(1)	number of lanes, 1: 1; 2: 2 or 3; 3: >=4

5.2 Compared Methods

In this section, we compare our proposed model with the following methods.

- **MultiLayer Perceptron (MLP):** This is a class of feedforward **Artificial Neural Network (ANN)**. An MLP consists of at least three layers of nodes: an input layer, a hidden layer, and an output layer. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training.
- **Seq2Seq** [48]**:** This model uses one RNN to encode the input sequences into a feature representation and another RNN to generate predictions.
- **GRU** [49]**: Gated Recurrent Units (GRUs)** are used as a gating mechanism in RNNs. A GRU is similar to an LSTM unit in that it has a forget gate, but it has fewer parameters than an LSTM unit because it lacks an output gate. GRUs have been shown to exhibit even better performance on certain datasets.
- **DCRNN** [9]**:** DCRNN is a deep learning framework for traffic forecasting that incorporates both the spatial and temporal dependencies of traffic flows. DCRNN captures spatial dependencies by means of bidirectional random walks on the graph and temporal dependencies by means of an encoder-decoder architecture with scheduled sampling.
- **STGCN** [8]**:** STGCN is a novel deep learning framework for traffic speed prediction that integrates graph convolution and gated temporal convolution through spatio-temporal convolutional blocks.
- **Graph WaveNet** [50]**:** Graph WaveNet is an effective method to learn hidden spatial dependencies automatically from the data; what's more, it can capture spatial-temporal

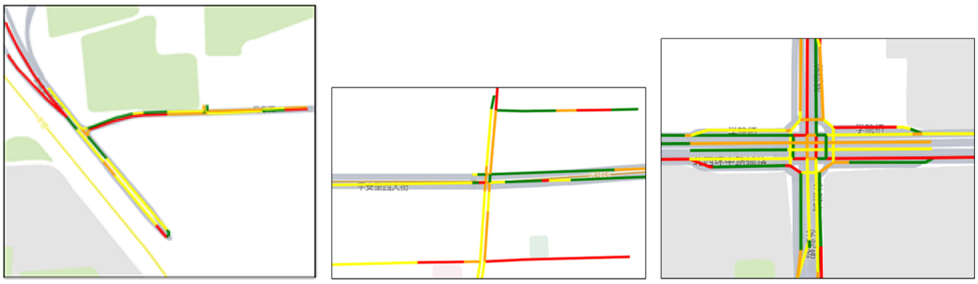

(a) Yue Quan Road (b) Ping An Li (c) Xue Yuan Road

Fig. 4. Three selected intersections.

dependencies efficiently and effectively by combining graph convolution with dilated casual convolution.

- **MTGNN** [51]**:** A novel mix-hop propagation layer and a dilated inception layer are further proposed to capture the spatial and temporal dependencies within the time series. The graph learning, graph convolution, and temporal convolution modules are jointly learned in an end-to-end framework.
- **AGCRN** [52]**:** AGCRN is a model that can capture node-specific spatial and temporal correlations in time-series data automatically without a pre-defined graph. It is proposed to enhance the traditional graph convolutional network with node adaptive parameter learning and data-adaptive graph generation modules for learning node-specific patterns and discovering spatial correlations from data, separately.

5.3 Evaluation Metrics

The **Mean Absolute Error (MAE)**, **Mean Absolute Percentage Error (MAPE)**, and **Root Mean Square Error (RMSE)** are used to evaluate the performance of the proposed method and the approaches considered for comparison. These metrics are defined as follows:

$$MAE = \frac{1}{T}\sum_{t=1}^{T}|S_t - \tilde{S}_t|, \tag{10}$$

$$MAPE = \frac{1}{T}\sum_{t=1}^{T}\left|\frac{S_t - \tilde{S}_t}{S_t}\right|, \tag{11}$$

$$RMSE = \sqrt{\frac{1}{T}\sum_{t=1}^{T}(S_t - \tilde{S}_t)^2}, \tag{12}$$

where S_t and $\tilde{S}_t$ are the real and predicted traffic speeds, respectively, at time t.

5.4 Implementation Details

We select three road intersections, Yue Quan Road, Ping An Li, and Xue Yuan Road, from the road network sub-dataset for the experiments, as shown in Figure 4. Specifically, the segments of these three roads are 14, 48, and 76 separately, which means the nodes in the linkage graph are the same size correspondingly and the connections between these segments are constructed based on the road map. Due to the complex relationships among the road sections involved, it is difficult to manually mark the connection types for all sections, so this attribute field is directly set as empty.

Consequently, in this article, only the driving relationships between sections as represented by the topological structure of the graph are used, and no complex edge properties are used.

In the Q-Traffic data from Baidu, 15 minutes is used as the length of a time slice (that is, 1 hour is divided into four time slices); accordingly, a day is divided into $24 \times 4 = 96$ time slices. In this article, we study long-term prediction and set the lengths of both the input and output sequences to 24, corresponding to a time length of 6 hours. In other words, our first input sequence is composed of the segment speeds collected between the first time slice (0:00–0:15) and the 24th time slice (5:45–6:00) on a certain day. Accordingly, the first output sequence is the following 24 time slices, that is, the speed sequence for the corresponding segment from the 25th time slice (6:00–6:15) to the 48th time slice (11:45–12:00). Subsequently, a second set of input (time slices from 2 to 25) and output (time slices from 26 to 49) sequences are generated by sliding the window, and so on. When the right edge of the sliding window corresponding to the output sequence reaches the last time slice of the day (namely, the 96th time slice, corresponding to 23:45–24:00), the window stops sliding, and all input and output sequences for the day have been generated. It is not difficult to calculate that the number of input sequences, and equivalently, the number of output sequences, corresponding to a day is $96 - 24 = 72$. The data from the most recent 7 days were used as the test set, the data from the previous 7 days were used as the verification set, and all remaining data were used as the training set. Because the Q-Traffic dataset covers 61 days, the numbers of days of data available for the training set, verification set, and test set are 47 days, 7 days, and 7 days, respectively.

In our experiments, the MSE loss was chosen as the loss function to train the models. In the MLP model, a total of five hidden layers were used, and the numbers of nodes contained in the hidden layers were 256, 128, 128, 64, and 64. *Leaky_ReLU* was selected as the activation function, 1,024 as the batch size, and 0.001 as the learning rate. In the Seq2Seq model, single-layer LSTM units were used as the neurons constituting the RNN, the length of each hidden state vector was set to 32, the batch size was selected to be 1,024, and the learning rate was 0.001. In the DCRNN, STGCN, Graph WaveNet, MTGNN, and AGCRN model, we download the open-sourced code and prepared our own dataset; the default parameters were used while the parameters that related to the data were changed to be consistent with our data. In our proposed model, we set layer number to 3 and number block size to 1. Due to the limitation of computing resources, we reduce the batch size to 256. The learning rate was chosen to be 0.001, while we adopt AdamOptimizer to optimize the model. The early stop mechanism was also used in the training process.

Actually, the design of our mixed attention mechanism is based on alternating propagation of vertical and horizontal attention. Information transmission based on vertical attention is denoted by V, while information transmission based on horizontal attention is denoted by H. Accordingly, the alternating transmission scheme can be expressed as "$V - H - V - H$," which is only one possible option; we can also choose other schemes, such as "$V - V - V - H$," "$V - H - H - H$," "$V - V - H - H$," or other non-alternating modes of information transmission (here, we take four instances of information transmission as the basis for our examples). We choose an experimental scenario of long-term prediction to investigate different transmission schemes. Mixed attention involves many hyper-parameters, and it is impossible to enumerate all possible combinations to find the optimal combination of hyper-parameters. In such cases, it is common to empirically specify some of the intuitively less important hyper-parameters and then specify the important hyper-parameters step by step in a greedy way. In our mixed attention model, we fix the hyper-parameters in accordance with our experience, including the information transmission process. The length of each hidden state vector was set to 8, the dropout ratio in the transverse attention stage was 0.05, the batch size for training was 1,024, and the learning rate was 0.001. The influence of the order of information transmission on the model prediction accuracy is shown in Figure 5. The results show that the prediction accuracy with "$V - V - V - H$" is the highest, while the

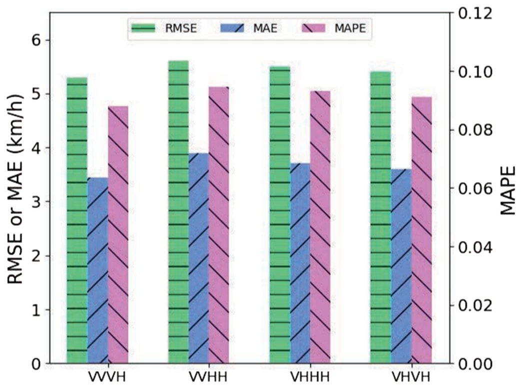


Fig. 5. Influence of the order of attention-based information transmission on the prediction accuracy.

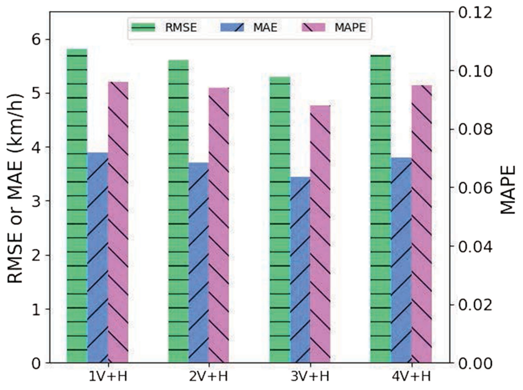


Fig. 6. Influence of the number of vertical attention layers on the prediction accuracy.

prediction accuracy with "$V-H-V-H$" is slightly lower but higher than that with the other two schemes. Based on these findings, we may guess that in our case, vertical attention is more important than horizontal attention; thus, horizontal attention information needs to be transmitted only once, and the number of instances of vertical attention information transmission can be increased appropriately. In addition, the prediction accuracy with "$V-H-V-H$" is higher than that with "$V-V-H-H$," indicating that the alternating mode of information transmission is better than stacking the two kinds of attention separately.

On the basis of the above experiments, we can guess that a relatively efficient means of information transmission is to combine multiple instances of vertical attention transmission with a single instance of horizontal attention transmission; this can be expressed as "$kV+H$," where k represents the number of vertical attention layers. Accordingly, we experimentally compare the predictive accuracy achieved with different numbers of horizontal attention layers. We compare the effects of models with values of k ranging from 1 to 4 and found that the prediction accuracy was the highest when k was set to 3, that is, when three horizontal attention layers were combined with one vertical attention layer. The corresponding results are shown in Figure 6. On this basis, for our experiments, we choose "$3V+H$" as the main scheme of GSeqAtt.

5.5 Ablation Study

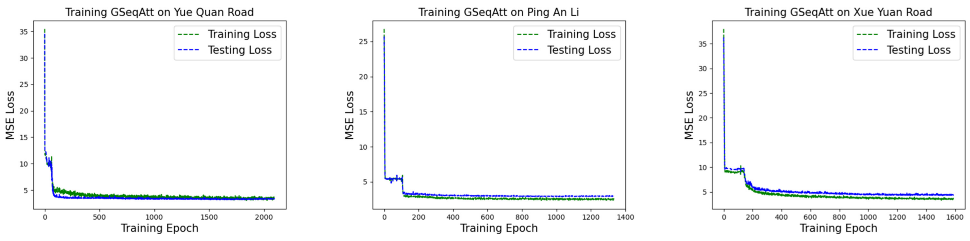
In our model, we utilize two kinds of attention mechanisms, namely, horizontal and vertical attention mechanisms. In order to check the effectiveness of each attention mechanism, we choose

Table 3. Comparison with Various Baselines

Data	Model	1 Hour			3 Hour			6 Hour		
		MAE	MAPE	RMSE	MAE	MAPE	RMSE	MAE	MAPE	RMSE
Yue Quan Road	MLP	3.4639	0.0761	5.2033	3.8042	0.0852	5.3786	3.8714	0.0954	5.7589
	Seq2Seq	3.3441	0.0712	4.9605	3.7053	0.0844	5.3310	3.8398	0.0936	5.7815
	GRU	3.6023	0.0782	5.3816	3.8271	0.0873	5.4769	3.9036	0.0940	5.7829
	DCRNN	3.1801	0.0683	4.7700	3.6107	0.0836	5.2622	3.6679	0.0887	5.5405
	STGCN	3.7518	0.0928	5.8262	4.3161	0.1025	6.0208	4.5795	0.1077	6.9070
	Graph WaveNet	3.6133	0.0835	5.5938	3.8367	0.0939	5.6311	3.9329	0.0958	6.1191
	MTGNN	7.7003	0.1849	10.6605	10.6500	0.2539	14.2389	11.4670	0.2736	14.8771
	AGCRN	7.3494	0.1742	11.2873	8.7907	0.1862	11.7982	10.5558	0.2570	13.9753
	VertiAtt	3.3328	0.0704	4.7607	3.6382	0.0811	5.1676	3.6543	0.0884	5.4415
	HoriAtt	3.0470	0.0683	4.5744	3.5939	0.0799	5.1474	3.6325	0.0879	5.4032
	GSeqAtt	**2.8914**	**0.0671**	**4.5623**	**3.2231**	**0.0776**	**5.0533**	**3.5711**	**0.0873**	**5.3287**
Ping An Li	MLP	2.3964	0.0918	3.5636	2.6957	0.1062	4.0639	3.0638	0.1219	4.4889
	Seq2Seq	2.3019	0.0885	3.5112	2.6641	0.1056	4.0309	2.8125	0.1129	4.2554
	GRU	2.3545	0.0898	3.5392	2.6742	0.1061	4.0333	2.8480	0.1127	4.2570
	DCRNN	2.2722	0.0876	3.4725	2.6409	0.1035	3.9951	2.7202	0.1146	4.2474
	STGCN	2.6922	0.1010	4.0085	2.8591	0.1108	4.2573	3.1295	0.1161	4.5530
	Graph WaveNet	2.3997	0.0943	3.6222	2.7357	0.1069	4.2076	3.7439	0.1345	4.5576
	MTGNN	3.5532	0.1469	5.3652	4.9791	0.2017	6.7455	5.2471	0.2126	7.0226
	AGCRN	3.4418	0.1274	5.0347	4.8049	0.1748	6.5659	5.1134	0.1865	6.8276
	VertiAtt	2.2456	0.0860	3.4508	2.3968	0.0892	3.5908	2.4516	0.0911	3.6027
	HoriAtt	1.9147	0.0735	3.1218	2.3525	0.0851	3.4376	2.3827	0.0873	3.5733
	GSeqAtt	**1.8871**	**0.0700**	**2.9792**	**2.2011**	**0.0799**	**3.3431**	**2.2117**	**0.0843**	**3.4183**
Xue Yuan Road	MLP	4.9368	0.1579	7.2483	4.9726	0.1611	7.3456	5.8719	0.1777	7.8787
	Seq2Seq	3.0370	0.0899	4.3218	4.2041	0.1298	5.9677	4.8374	0.1542	6.9035
	GRU	3.1019	0.0907	4.4120	4.2916	0.1373	6.0957	5.0530	0.1641	7.1804
	DCRNN	2.7982	0.0861	4.1683	3.6573	0.1101	5.3324	3.7953	0.1151	5.6461
	STGCN	2.8467	0.0876	4.2055	4.0147	0.1208	5.9188	4.7712	0.1277	6.7191
	Graph WaveNet	3.3635	0.1014	4.5819	4.4054	0.1376	6.2891	4.9939	0.1626	7.3846
	MTGNN	7.5150	0.2211	9.7463	7.7815	0.2345	10.8573	8.8588	0.2679	11.6632
	AGCRN	7.2704	0.1995	8.9715	7.7378	0.2209	10.1259	8.4098	0.2399	11.3825
	VertiAtt	2.7698	0.0818	4.0068	3.4129	0.1074	5.1399	3.6051	0.1104	5.4934
	HoriAtt	2.6293	0.0762	3.8756	3.1697	0.0982	5.1001	3.4064	0.1053	5.2924
	GSeqAtt	**2.4062**	**0.0705**	**3.6453**	**2.9817**	**0.0879**	**4.3997**	**3.2312**	**0.0953**	**4.6851**

to remove one component for testing. For checking the vertical attention mechanism, we name HoriAtt, which contains vanilla GNN with default aggregation function and Transformer, to do experiments. Actually, HoriAtt was named as GTransformer before in this article, which is constructed based on an attention mode, and it can process a graph sequence structure. Meanwhile, VertiAtt is a new baseline for checking that the horizontal attention mechanism actually contributes to the final performance. In VertiAtt, we combine GNAtt, which modifies the original aggregation function, and seq2seq to capture graph sequence input.

For this purpose, we design an experimental ablation study based on the selected roads shown in Figure 4. In Table 3, we compare the results of the two variants of our model. We can find that HoriAtt achieves better performance than VertiAtt on different road segments. It means that the Transformer mechanism contributes much more than the attention mechanism inside a GN block due to the multi-head attention in capturing dependencies between the input sequence. GSeqAtt shows better performance than HoriAtt, indicating that the vertical attention mechanism indeed improves the prediction performance. In other words, we can conclude that assigning different levels of importance to related edges during the node update process in a GN block is necessary.



(a) Yue Quan Road (b) Ping An Li (c) Xue Yuan Road

Fig. 7. Training losses on the three selected road intersections.

5.6 Results and Discussion

In Table 3, we present the comparison with the results of the baseline models for the three selected roads in Beijing in terms of the three evaluation metrics. We find that the same methods trained on different data exhibit different performances. Specifically, the models trained on Xue Yuan Road yield worse results due to the higher complexity of the road segments, while better results are obtained for the other two roads. From this table, we can easily observe that the proposed GSeqAtt method outperforms simple deep learning methods such as MLP, Seq2Seq, and GRU. These simple methods cannot utilize road topology information and do not capture the spatial dependencies of the road network. In contrast, the GSeqAtt model can capture the graph structure and utilize latent properties to model temporal and spatial dependencies to improve performance. Furthermore, we also consider state-of-the-art methods, such as DCRNN, STGCN, Graph WaveNet, MTGNN, and AGCRN; all of these models construct a graph as an adjacent matrix to characterize the internal properties of the graph. Nevertheless, the results in Table 3 show that these methods also cannot model the graph structure and achieve worse performance than GSeqAtt, which considers the link directions and models the road network as a directed graph. We find several interesting things: on different road segments, the models achieve different performance. For example, the MLP can achieve better performance than GRU in Yue Quan Road while getting the opposite results when in the other two roads. What's more, DCRNN can get the best performance on different roads when compared with other baselines. Compared with different forecasting periods, we can find that the performance of all methods will be worse than before when we enlarge the forecasting future period.

We also present the training process in terms of the training and testing losses of our proposed model on the three selected roads in Figure 7, which shows that these losses start at a high level and then decrease to lower levels until convergence is reached. The high initial loss indicates that the performance is poor when training begins. After a sufficient number of iterations, the MSE loss reaches a relatively low level, indicating an acceptable result. Moreover, the forecasting accuracy of GSeqAtt is consistently superior to that of HoriAtt. Thus, we can conclude that GSeqAtt is more suitable for long-term forecasting tasks.

In addition to helping to increase the training efficiency and improve the prediction accuracy of the model, the attention mechanism also plays an important role in measuring the importance of different types of contextual information gathered for an element. Taking vertical attention as an example, for a road network with 14 nodes and 26 edges, Figure 8 shows heat maps representing different attention-based weights measured for the transmission of edge information to nodes. It can be seen from this figure that the heat map is relatively sparse because in a GN block (or GNAtt), information propagates to nodes only from adjacent edges, whereas there is no information propagation between non-adjacent edges and nodes. Second, information flowing to different sides of the same node is given different weights, indicating that the attention mechanism plays a

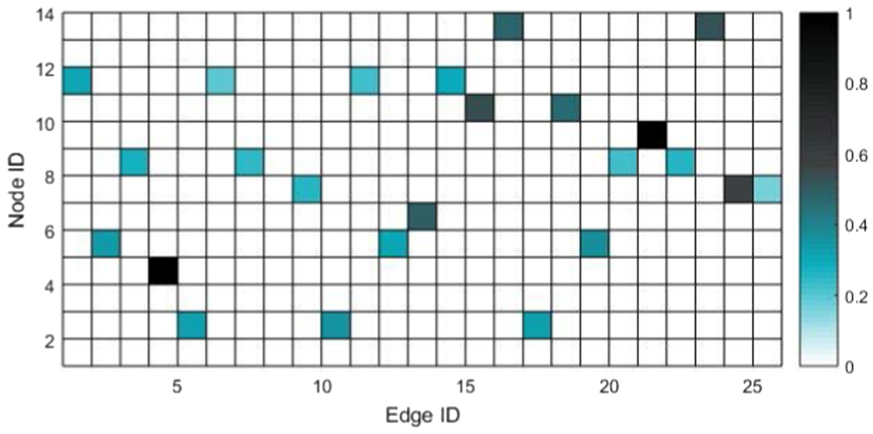


Fig. 8. Different weights of information transmission in vertical attention.

role in differentiating the disseminated information. Finally, unlike a GAT, GNAtt does not involve self-propagation of information because information in GNAtt flows from edge to node, whereas information in a GAT flows from node to node.

6 CONCLUSION

In this article, we study how to combine an attention mechanism with a GNN for capturing the information contained in graph sequences to improve prediction accuracy. We propose a novel mixed attention-based GNN model named GSeqAtt. Specifically, we design a new block structure named GNAtt that incorporates an attention mechanism by revising the update procedure such that different levels of importance are assigned to related edges when updating nodes. To capture the relations among the graphs in the input sequence, we propose the GTransformer model, which adopts the self-attention mechanism of the Transformer model. With these two attention mechanisms, we can not only process time series to capture the correlations between graphs in the input time sequences by means of a horizontal attention mechanism but also capture information contained within the graph structure when updating node information by considering edge importance in each frame of the time series by means of a vertical attention mechanism. Then, we conduct extensive experiments on real-world data from three selected roads in Beijing for the traffic speed prediction task. The results demonstrated the effectiveness and robustness of the proposed model.

In future work, we will investigate the influence of other external factors, such as the weather and traffic incidents, to improve the prediction accuracy. More variants of attention mechanisms can also be combined with GNNs, which may be a new direction for future work. The time and memory requirements of such models are still two major problems that should be addressed in future work. It would also be meaningful to explore the explainability of GNNs. Actually, it takes a long time to start the training process when running the GNN models because of loading the whole graph; in the future, we will study how to reduce the size of the trained graph and utilize the subgraph or mini graph to improve the efficiency of the training process.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their constructive comments on this research work.

REFERENCES

[1] Junping Zhang, Fei-Yue Wang, Kunfeng Wang, Wei-Hua Lin, Xin Xu, and Cheng Chen. 2011. Data-driven intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems* 12, 4 (2011), 1624–1639.

[2] Chun-Hsin Wu, Jan-Ming Ho, and Der-Tsai Lee. 2004. Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems* 5, 4 (2004), 276–281.

[3] Billy M. Williams and Lester A. Hoel. 2003. Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results. *Journal of Transportation Engineering* 129, 6 (2003), 664–672.

[4] Xiaolei Ma, Zhimin Tao, Yinhai Wang, Haiyang Yu, and Yunpeng Wang. 2015. Long short-term memory neural network for traffic speed prediction using remote microwave sensor data. *Transportation Research Part C: Emerging Technologies* 54 (2015), 187–197.

[5] Xiaolei Ma, Zhuang Dai, Zhengbing He, Jihui Ma, Yong Wang, and Yunpeng Wang. 2017. Learning traffic as images: A deep convolutional neural network for large-scale transportation network speed prediction. *Sensors* 17, 4 (2017), 818.

[6] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 1655–1661.

[7] Dong Wang, Junbo Zhang, Wei Cao, Jian Li, and Yu Zheng. 2018. When will you arrive? Estimating travel time based on deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 1–8.

[8] Bing Yu, Haoteng Yin, and Zhanxing Zhu. 2018. Spatio-temporal graph convolutional Networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence.* 3634–3640.

[9] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations.*

[10] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S. Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems* 1 (2020), 4–24.

[11] Franco Scarselli, Ah Chung Tsoi, Marco Gori, and Markus Hagenbuchner. 2004. Graphical-based learning environments for pattern recognition. In *Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR).* 42–56.

[12] Ozan Irsoy and Claire Cardie. 2014. Deep recursive neural networks for compositionality in language. In *Advances in Neural Information Processing Systems.* 2096–2104.

[13] Thomas N. Kipf and Max Welling. 2017. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations.*

[14] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention Networks. In *International Conference on Learning Representations* (accepted as poster).

[15] Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. 2018. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261* (2018).

[16] Volodymyr Mnih, Nicolas Heess, Alex Graves, et al. 2014. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems.* 2204–2212.

[17] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning.* 2048–2057.

[18] Hongyang Gao, Zhengyang Wang, and Shuiwang Ji. 2020. Kronecker attention networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 229–237.

[19] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations (ICLR'15), Conference Track Proceedings*, Yoshua Bengio and Yann LeCun (Eds.).

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems.* 5998–6008.

[21] J. Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and D. Yeung. 2018. GaAN: Gated attention networks for learning on large and spatiotemporal graphs. In *Proceedings of the 34th Conference on Uncertainty in Artificial Intelligence.* 339–349.

[22] Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 922–929.

[23] Chuanpan Zheng, Xiaoliang Fan, Cheng Wang, and Jianzhong Qi. 2020. Gman: A graph multi-attention network for traffic prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence.* 1234–1241.

[24] Haoxing Lin, Rufan Bai, Weijia Jia, Xinyu Yang, and Yongjian You. 2020. Preserving dynamic attention for long-term spatial-temporal prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 36–46.

[25] Chang Guo, Demin Li, Guanglin Zhang, and Menglin Zhai. 2018. Real-time path planning in urban area via vanet-assisted traffic information sharing. *IEEE Transactions on Vehicular Technology* 67, 7 (2018), 5635–5649.

[26] Nancy L. Nihan and Kjell O. Holmesland. 1980. Use of the box and Jenkins time series technique in traffic forecasting. *Transportation* 9, 2 (1980), 125–143.

[27] Yudong Chen, Yi Zhang, and Jianming Hu. 2008. Multi-dimensional traffic flow time series analysis with self-organizing maps. *Tsinghua Science & Technology* 13, 2 (2008), 220–228.

[28] Bidisha Ghosh, Biswajit Basu, and Margaret O'Mahony. 2009. Multivariate short-term traffic flow forecasting using time-series analysis. *IEEE Transactions on Intelligent Transportation Systems* 10, 2 (2009), 246.

[29] Shan-Huen Huang and Bin Ran. 2003. *An Application of Neural Network on Traffic Speed Prediction under Adverse Weather Condition*. Ph.D. Dissertation. University of Wisconsin–Madison.

[30] Alireza Khotanzad and Nayyara Sadek. 2003. Multi-scale high-speed network traffic prediction using combination of neural networks. In *Proceedings of the International Joint Conference on Neural Networks, 2003.* Vol. 2. IEEE, 1071–1075.

[31] Jingyuan Wang, Qian Gu, Junjie Wu, Guannan Liu, and Zhang Xiong. 2016. Traffic speed prediction and congestion source exploration: A deep learning method. In *2016 IEEE 16th International Conference on Data Mining*. 499–508.

[32] Junbo Zhang, Yu Zheng, and Dekang Qi. 2017. Deep spatio-temporal residual networks for citywide crowd flows prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 1655–1661.

[33] Youngjoo Kim, Peng Wang, Yifei Zhu, and Lyudmila Mihaylova. 2018. A capsule network for traffic speed prediction in complex road Networks. In *2018 Sensor Data Fusion: Trends, Solutions, Applications*. 1–6.

[34] Yuxuan Liang, Songyu Ke, Junbo Zhang, Xiuwen Yi, and Yu Zheng. 2018. Geoman: Multi-level attention networks for geo-sensory time series prediction. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. 3428–3434.

[35] Youngjoo Kim, Peng Wang, and Lyudmila Mihaylova. 2019. Structural recurrent neural network for traffic speed prediction. In *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'19)*. 5207–5211.

[36] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and M. Sun. 2018. Graph Neural Networks: A review of methods and applications. *ArXiv* abs/1812.08434 (2018).

[37] Franco Scarselli, Sweah Liang Yong, Marco Gori, Markus Hagenbuchner, Ah Chung Tsoi, and Marco Maggini. 2005. Graph neural networks for ranking web pages. In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence*. 666–672.

[38] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks* 20, 1 (2008), 81–102.

[39] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2008. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2008), 61–80.

[40] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. 2016. Gated graph sequence neural networks. In *4th International Conference on Learning Representations (ICLR'16), Conference Track Proceedings*.

[41] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. In *2nd International Conference on Learning Representations (ICLR'14), Conference Track Proceedings*.

[42] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. 2017. Neural message passing for quantum chemistry, In *Proceedings of Machine Learning Research*, Doina Precup and Yee Whye Teh (Eds.), Vol. 70. 1263–1272. http://proceedings.mlr.press/v70/gilmer17a.html.

[43] Zhiyong Cui, Kristian Henrickson, Ruimin Ke, and Yinhai Wang. 2019. Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting. *IEEE Transactions on Intelligent Transportation Systems* 21, 11 (2019), 4883–4894.

[44] Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. 1412–1421.

[45] Matthew Veres and Medhat Moussa. 2019. Deep learning for intelligent transportation systems: A survey of emerging trends. *IEEE Transactions on Intelligent Transportation Systems* (2019), 24 pages.

[46] Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S. Yu. 2019. Heterogeneous graph attention network. In *The World Wide Web Conference*. 2022–2032.

[47] Xiaoyu Wang, Cailian Chen, Yang Min, Jianping He, Bo Yang, and Yang Zhang. 2018. Efficient metropolitan traffic prediction based on graph recurrent neural Network. *arXiv preprint arXiv:1811.00740* (2018).

[48] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems*. 3104–3112.

[49] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*.

[50] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. 1907–1913.

[51] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, Xiaojun Chang, and Chengqi Zhang. 2020. Connecting the dots: Multivariate time series forecasting with graph neural networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 753–763.

[52] Lei Bai, Lina Yao, Can Li, Xianzhi Wang, and Can Wang. 2020. Adaptive graph convolutional recurrent network for traffic forecasting. In *Advances in Neural Information Processing Systems*.

Received November 2020; revised May 2021; accepted June 2021