TraverseNet: Unifying Space and Time in Message Passing for Traffic Forecasting

Zonghan Wu, Da Zheng, Shirui Pan, Quan Gan, Guodong Long, and George Karypis, Fellow, IEEE

Abstract—This article aims to unify spatial dependency and temporal dependency in a non-Euclidean space while capturing the inner spatial–temporal dependencies for traffic data. For spatial–temporal attribute entities with topological structure, the space–time is consecutive and unified while each node’s current status is influenced by its neighbors’ past states over variant periods of each neighbor. Most spatial–temporal neural networks for traffic forecasting study spatial dependency and temporal correlation separately in processing, gravely impaired the spatial–temporal integrity, and ignore the fact that the neighbors’ temporal dependency period for a node can be delayed and dynamic. To model this actual condition, we propose TraverseNet, a novel spatial–temporal graph neural network, viewing space and time as an inseparable whole, to mine spatial–temporal graphs while exploiting the evolving spatial–temporal dependencies for each node via message traverse mechanisms. Experiments with ablation and parameter studies have validated the effectiveness of the proposed TraverseNet, and the detailed implementation can be found from https://github.com/nnzhan/TraverseNet.

Index Terms—Deep learning, graph autoencoder, graph convolutional networks, graph neural networks (GNNs), graph representation learning, network embedding.

I. INTRODUCTION

Spatial–temporal graph neural networks (STGNNs) for traffic forecasting is a new research topic that emerged very recently. It involves message passing in a dynamic system in which node features are constantly changing over time. STGNNs assume that the nodes in a topological structure contain spatial–temporal dependencies where a node’s future pattern is subject to its neighbors’ historical results as well as its own past records. Specifically, the traffic conditions on a particular road at a given time depend not only on that road’s previous traffic conditions, but also on the traffic conditions of its adjacent roads several minutes ago, as it takes time for vehicles to travel from one location to the next. Hence, how to effectively exploit and preserve both spatial and temporal inner-dependency turns into an essential challenge to answer.

Existing graph neural networks (GNNs) or graph convolutional neural networks (GCNs) cannot solve this problem alone. The main reason is that they are designed to handle static node features so that only spatial dependencies can be captured. To jointly capture spatial topology and temporal sequence in a spatial–temporal graph (ST-Graph), researchers naturally considered combining GCNs and recurrent neural networks (RNNs) [1]–[3]. While effective...
in fusing topological information into temporal sequence learning, RNN-based frameworks are inefficient in capturing long-range spatial–temporal dependencies. As illustrated in Fig. 1(a), during each recurrent step, they only allow a node to be aware of its neighbors’ current inputs and its neighbors’ previous hidden states. Information loss is inevitable when processing long sequences.

Meanwhile, capitalizing on parallel computing and stable convolutional propagation, convolutional neural network (CNN)-based ST-Graph frameworks have received considerable attention [4], [5]. They stack temporal convolution layers and graph convolution layers to capture local spatial–temporal dependencies. As illustrated in Fig. 1(a), they first use a small 1-D convolutional kernel to propagate nodes’ temporal information to the current time step and then use the graph convolution to pass the aggregated temporal information of a node’s neighbors to the node itself. To capture the long-range spatial–temporal dependencies, they face the tradeoff between kernel size and the number of layers. If a large 1-D kernel is used to retain long-range spatial–temporal relations, the model has to be shallow because of the small number of layers. Alternatively, if a small kernel is used, a large number of 1-D CNN layers and graph convolutional layers are requested, resulting in efficiency issues.

The attention mechanism is known for its efficiency in delivering important information [6], [7]. A number of studies integrate spatial attention with temporal attention for spatial–temporal data modeling [8]–[10]. The information flow diagram of attention-based methods is similar to that of CNN-based methods. Instead of using convolutional kernels, they apply attention mechanisms for information aggregation. As illustrated in Fig. 1(c), they first use temporal attention to pass important historical information of each node to its current step. Then they aggregate neighborhood information selectively for each node by spatial attention. In this way, each node will receive its neighbor’s temporal information indirectly. However, a node cannot determine which period of temporal information from the neighborhood is more relevant to itself because the computation of a neighbor’s temporal representation is independent of the central node itself.

To summarize, existing approaches either model spatial–temporal dependencies locally or model spatial correlations and temporal correlations separately. They prevent a node from being directly aware of its neighborhood’s long-range historical information. In fact, a node’s current state may depend on its neighbors’ previous states within a certain period of time. As illustrated in Fig. 1(d), the rise of a node’s curve may exert influence on its neighbors several time steps later because of physical distance. For example, traffic congestion on a road will cause another congestion on its nearby roads 15 min later. It suggests that treating spatial correlations and temporal correlations locally or separately is inappropriate. In fact, the spatial–temporal dependency is a whole that cannot be separated into the spatial dependency and the temporal dependency. (We refer to this as spatial–temporal integrity.) Additionally, existing ST-GNNs tend to simply stack different layers (e.g., inception layer, dilated convolutional layer, and RNN/CNN layer) together, resulting in overly complex and cumbersome architecture. Such construction confuses the significance and contribution of each kind of layer.

To overcome the above challenges, we present TraverseNet, a novel spatial–temporal neural network for structured data. TraverseNet processes a spatial–temporal graph as an inseparable entity. Our specially designed message traverse layer enables a node to be wise to a period of information from its neighborhood explicitly. We leverage attention mechanisms to select influential neighboring conditions of a node. Instead of attending the central node’s current state with its neighbors’ concurrent states, we attend a node’s current state over each of its neighbors’ historical states within a certain period of time. By building connections from each neighbor’s past to the central node’s present, a node’s neighborhood information no matter in the past or the present is traversed efficiently and effectively.

The main contributions of this article are as follows:

1) We propose TraverseNet, a simple and powerful framework that captures the inner spatial–temporal dependencies without compromising spatial–temporal integrity.

2) We propose a message traverse layer, effectively unifying space and time in message passing by traversing information of a node’s neighbors’ past to the node’s present.

3) We construct TraverseNet with message traverse layers and validate the significance of the message traverse mechanism with an experimental study.

II. BACKGROUND AND RELATED WORK

A. Definitions and Notations

An attributed graph is defined as \( G = (V, E, X) \), where \( V \) is the set of nodes, \( E \) is the set of edges, and \( X \) is a node feature matrix. We let \( v \in V \) to represent a node and \( e = (v, u) \in E \) to denote an directed edge from \( u \) to \( v \). The neighborhood of a node \( v \) is the set of nodes \( N(v) = \{ u \in V | (v, u) \in E \} \) that points to node \( v \). The adjacency matrix \( A \) is a mathematical way that defines a graph with a sparse matrix. It is an \( N \) by \( N \) matrix with \( A_{ij} = 1 \) if \( (v_i, v_j) \in E \) and \( A_{ij} = 0 \) if \( (v_i, v_j) \notin E \), where \( N \) is the number of nodes. We define a spatial–temporal graph as a sequence of graphs that evolve over time, \( G_{h\rightarrow t} = \{ G_{h\rightarrow t_1}, G_{h\rightarrow t_2}, \ldots, G_{h\rightarrow t_n} \} \). In this article, we assume \( G_h = (V, E, X_h) \), where \( X_h \in \mathbb{R}^{N \times D} \) denote the node feature matrix at time step \( t \) and \( D \) is the feature dimension.

The goal of spatial–temporal graph forecasting is to predict future spatial–temporal graphs given its historical data. Formally, the spatial–temporal graph forecasting problem is defined as finding a mapping from the historical values to the future values

\[
G_{h\rightarrow t} \rightarrow G_{h\rightarrow t+1\rightarrow t+\ell}.
\]

The major notations in this article are listed in Table I.

B. Graph Neural Networks (GNNs)

GNNs extract high-level representations of nodes for graphs by graph convolution [11], [12], [13], or message passing [14],
TABLE I

| Notations | Descriptions |
|-----------|-------------|
| $G$       | A graph.    |
| $V$       | The set of nodes in a graph. |
| $v$       | A node $v \in V$. |
| $E$       | The set of edges in a graph. |
| $N(v)$    | The neighbors of a node $v$. |
| $\Lambda$ | The graph adjacency matrix. |
| $X$       | The feature matrix of a graph. |
| $X_t$     | The node feature matrix of a graph at the time step $t$. |
| $H$       | The hidden feature matrix. |
| $h_v$     | The hidden feature vector of node $v$. |
| $W$, $\Theta$, $U$, $\gamma$ | Learnable model parameters. |

[15] in (semi)supervised or self-supervised learning manners [16]–[19]. Graph convolution methods are based on the eigendecomposition of the graph Laplacian matrix. The motivation of graph convolution methods is to remove noise from graph signals. Beyond that, many graph convolution methods can be interpreted from the perspective of message passing. The form of recurrent-based approaches can be conceptualized as

$$H_t = \text{RNN}(\text{GCN}(X_t, H_{t-1}), A; \Theta); U)$$

where $\Theta$ and $U$ are model parameters, and $H_{t-1}$ represents the nodes’ previous hidden state matrix. The previous hidden state of a node is essentially a memory vector of its historical information. As the number of recurrent steps increases, the memory vector will gradually forget information many steps before. Seo et al. [1] are the first to propose a recurrent-based STGNN. They adopt the long short-term memory networks (LSTMs) as the RNN component and ChebNet as the GCN component. Li et al. [2] consider information diffusion in designing their framework. They replace LSTMs with gated recurrent units (GRUs) and propose diffusion graph convolution. Zhang et al. [3] further propose multihead graph attention as the GCN part. Considering node data distributions may differ, Pan et al. [26] introduce a hypernetwork that is able to generate a set of STGNN model parameters for each node based on its static node features. The common drawbacks of recurrent-based STGNNs are the high computation cost and gradient diminishing problem induced by recurrent propagation accompanied by graph convolution.

2) Convolution-Based STGNNs: Convolution-based STGNNs take advantage of the efficiency and shift-invariance property of convolutional neural networks [27]–[30]. They interleave temporal convolutions with graph convolutions to handle temporal correlations and spatial dependencies, respectively. The core difference to recurrent-based methods is that they replace RNNs with temporal convolution networks (TCNs) for capturing temporal patterns, as illustrated in the following:

$$H = \text{GCN}(\text{TCN}([x^T_{t=1}], \Theta), A; U)$$

where $\| \|$ represents concatenation, $H \in R^{N \times D \times (T-c+1)}$, $T$ is the sequence length, and $c$ is the kernel size of the temporal convolution. The information flow of an STGNN layer occurs first temporally and then spatially. Such models are against nature, where an object moves in space and time simultaneously. Li et al. [27] first propose a CNN-based STGNN. They adopt standard 1-D convolution to capture temporal dependencies and ChebNet to capture spatial dependencies. The use of standard 1-D convolution is inefficient to handle long-term dependencies. Wu et al. [28] propose Graph WaveNet that adopts dilated 1-D convolution. They further propose a self-adaptive graph learning module that can learn latent spatial dependencies from data. While Graph WaveNet only learns static graph structures, Zhang et al. [30] consider learning both static and dynamic, global and local graph structures.

3) Attention-Based STGNNs: Similar to convolution-based approaches, attention-based STGNNs treat spatial dependencies and temporal correlations in separate steps [8]–[10], [31]

$$H = \text{SA}([x^T_{t=1}, T A(X_t; \Theta), A]; U)$$

where $\text{SA}(\cdot)$ is a spatial attention layer and $TA(\cdot)$ is an temporal attention layer. The motivation of attention-based methods is that node spatial dependency and temporal dependency could be dynamically changing over time. The spatial attention layer first updates the graph adjacency matrix by computing the distance between a query node’s input and a key node’s input, then performing message passing. The temporal convolution layer computes a weighted sum of a node’s historical state based on attention scores. Guo et al. [8] propose

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
attention based spatial-temporal graph convolutional networks for traffic flow forecasting (ASTGCN) that interleaves spatial attention with temporal attention operations. To increase the scalability of spatial attention, Zheng et al. [31] propose hierarchical spatial attention that splits nodes into groups and performs intergroup attention and intragroup attention. Later works [9], [10] on attention-based STGNNs are not efficient to transfer the historical information of a node’s neighbor to the node itself due to local connections and fixed dependency weights. Most STSGCN and STFGNN are not efficient to transfer the historical information of a node’s neighbor to the node itself due to local connections and fixed dependency weights. Most recently, Pan et al. [34] and Hadou et al. [35] study stability of STGNNs. They design STGNNs that are stable to small perturbations in the underlying graphs. Pan et al. conduct experiments on human action recognition tasks [34]. Hadou et al. validate their method on flocking and motion planning [35]. Both of them do not consider traffic forecasting. Isufi et al. propose graph-time convolutional neural networks (GTCNN) based on graph product [36]. While enabling spatial-temporal dependencies, graph product is limited to fixed edge weights and undirected spatial-temporal relations.

In a broader context, our article and previously discussed related work only focus on predicting the trend of node features for spatial-temporal graphs. There are several research that conduct a more general study in which a full graph (nodes, edges, and attributes) is predicted [37], [38].

III. TraverseNet

To enable the direct flow of information both in space and time, we propose a novel STGNN named TraverseNet. The proposed design of TraverseNet is simple. Apart from multilayer perceptrons (MLPs), TraverseNet only contains our newly proposed message traverse layer. In the following, we introduce the message traverse layer and present the model framework of TraverseNet.

A. Message Traverse Layer

Static graph convolution layers or message passing layers only pass nodes’ neighborhood information across space, ignoring temporal dynamics. We propose the message traverse layer, a message passing layer that allows node neighborhood information to be simultaneously delivered across space and time. As demonstrated in Fig. 2, a node \( v \) has two neighbors \( u_1 \) and \( u_2 \). States of \( u_1 \) prior to \( t_4 \) can traverse to the state of node \( v \) at \( t_4 \) directly. This is in fundamental contrast to existing spatial-temporal works, where node \( v \) at \( t_4 \) can only receive information of node \( u_1 \) and \( u_2 \) from their concurrent time step \( t_4 \). The spatial-temporal dependency between two nodes may change from time to time depending on their states. We further leverage the attention mechanism to select important neighborhood information and to handle dynamic spatial-temporal dependencies. Formally, the message traverse layer updates the state of node \( v \) for each time step from \( t = 0 \) to \( t = p - 1 \) by

\[
\mathbf{h}^{(k)}_{v_t} = \sum_{u \in N(v) \cup \{v\}} \alpha^{(k)}_{v} (\mathbf{c}^{(k)}_{(u \rightarrow v)}, \mathbf{c}^{(k)}_{(u \rightarrow v)}) \mathbf{W}^{(k)}_{s} \mathbf{c}^{(k)}_{(u \rightarrow v)},
\]

where \( \mathbf{h}^{(0)}_{v_t} = \mathbf{x}_v \) and \( \mathbf{W}^{(k)}_{s} \) represent learnable model parameters at the \( k^{th} \) message traverse layer. The function \( \alpha^{(k)}_{v} (\cdot, \cdot) \) is an attention function of the form

\[
\alpha^{(k)}_{v} (z_v, z_u) = \frac{\exp\left(\sigma(\mathbf{r}_v \cdot \mathbf{q}^{(k)}(\mathbf{z}_v) \mathbf{\Theta}_c^{(k)} \mathbf{z}_u)\right)}{\sum_{m \in S} \exp\left(\sigma(\mathbf{r}_v \cdot \mathbf{q}^{(k)}(\mathbf{z}_v) \mathbf{\Theta}_c^{(k)} \mathbf{z}_m)\right)}
\]

where \( S = \{N(v) \cup \{v\}\} \), \( \mathbf{\Theta}_c \), \( \gamma \) represent model parameters, \( z_v \) and \( z_u \) denote the input node hidden features of the attention function, and \( \sigma(\cdot) \) denote an activation function. The term \( \mathbf{c}^{(k)}_{(u \rightarrow v)} \) represents the latent information received by node \( v \) from its own time steps prior to time \( t \). It is calculated as a weighted sum of its own historical states

\[
\mathbf{c}^{(k)}_{(u \rightarrow v)} = \sum_{m=0}^{Q} \alpha^{(k)}_{v} (\mathbf{h}^{(k-1)}_{u_m}, \mathbf{h}^{(k-1)}_{v_m}) \mathbf{W}^{(k)}_{c} \mathbf{c}^{(k)}_{(u_m \rightarrow v)},
\]

The term \( \mathbf{c}^{(k)}_{(u \rightarrow v)} \) denotes the latent information received by node \( v \) from its neighbor \( u \)’s previous time steps prior to time \( t \). To assess the importance of each neighbor \( u \), we involve the state of node \( v \) at the current time step as a query in the attention function

\[
\mathbf{c}^{(k)}_{(u \rightarrow v)} = \sum_{m=0}^{Q} \alpha^{(k)}_{v} (\mathbf{h}^{(k-1)}_{v_m}, \mathbf{h}^{(k-1)}_{v_m}) \mathbf{W}^{(k)}_{c} \mathbf{c}^{(k)}_{(u_m \rightarrow v)}.
\]

The attention functions \( \alpha^{(k)}_{v} (\cdot, \cdot) \) and \( \alpha^{(k)}_{v} (\cdot, \cdot) \) have the same form as \( \alpha^{(k)}_{v} (\cdot, \cdot) \). The hyperparameter \( Q \) controls the time-window size within which a node receives information from its neighbor’s past states. We differentiate the node itself from its neighborhood set because the node’s own information has a decisive influence on its predictions for sequence forecasting. The computation complexity of the proposed message traverse layer is \( O(M \times p \times Q) \), where \( M \) denotes the number of edges including self-loops and \( p \) is the input sequence length.

Our message traverse layer is a generalization of both spatial attention layers and temporal attention layers. Specifically,
if the neighborhood set of the node $v$ is empty, there will be only one term in the summation in (6) with the attention weight being one. Equation (5) reduces to a temporal attention layer

$$h^{(k)}_{v_i} = W^{(k)}_v \sum_{m=0}^Q a_{v_i}^{(k)} (h^{(k-1)}_{v_i}, h^{(k)}_{v_{m}}, h^{(k-1)}_{v_{m}}). \quad (9)$$

Similarly, if the window size $Q$ is set to 0, then (5) becomes a spatial attention layer

$$h^{(k)}_{v_i} = \sum_{u \in N(v) \cup v} a_{v_i}^{(k)} (W^{(k)}_v h^{(k-1)}_{v_i}, W^{(k)}_v h^{(k-1)}_{u}, W^{(k)}_v c^{(k)}_{i \rightarrow u}). \quad (10)$$

On the contrary to existing works that interleave spatial computations with temporal computations, the proposed message traverse layer handles spatial–temporal dependency as a whole. Our message traverse layer cannot be separated into a spatial attention layer and a temporal attention layer. It is mainly because we assume the spatial–temporal dependency between a node’s state at time step $t$ and its neighbor’s state at time step $t - m$ is dynamic. We use the state of the central node $v$ at time step $t$ as a query to access the importance of its neighboring node $u$’s historical states at each time step. Overall, this design shortens the path length of message passing and enables a node to be aware of its neighborhood variation at firsthand.

### B. Model Framework

As the message traverse layer is sufficient to capture spatial–temporal dependencies, we design a framework named TraverseNet that is simple and powerful to accomplish the spatial–temporal graph forecasting task. In Fig. 3, we present the framework of TraverseNet. The TraverseNet consists of three parts: the preprocessing layer, a stack of message traverse layers, and the postprocessing layer. The preprocessing layer is a feedforward layer that projects node feature matrix at each time step to a latent space. Next, we capture nodes’ spatial–temporal dependencies by message traverse layers. Finally, we use the postprocessing layer to map node hidden states to the output space. The postprocessing layer contains a $1 \times p$ standard convolutional layer followed by a feedforward layer. Suppose the input of node $v$ to the postprocessing layer is $z_v = \|_{i=0}^{p-1} h^{(K-1)}_{v_i}$, where $z_v \in \mathbb{R}^{d \times 1 \times p}$, and $p$ is the input sequence length. The $1 \times p$ standard convolutional layer is used to squeeze the third dimension of the input $z_v$ to 1. Afterward, the feedforward layer is applied to generate the prediction $z^*_{v} \in \mathbb{R}^d$ for node $v$, where $q$ is the output sequence length. In addition, residual connections and batch normalization are applied to message traverse layers to improve model robustness. In particular, as the input of each node may have very different scales, we let the batch normalization scale the hidden features on the node dimension.

#### C. Optimization & Implementation

We optimize model parameters of TraverseNet end-to-end by minimizing the mean absolute error (MAE) loss with gradient descent. The MAE is defined as

$$L = \text{Average} \left( \sum_{i=p+1}^{p+q} |X_{i} - \hat{X}_{i}| \right). \quad (11)$$

We implement TraverseNet with Pytorch and Deep Graph Library (DGL) [39]. In more detail, we first construct a heterogeneous graph by treating each node at each time step as a unique node and creating connections as illustrated in Fig. 2. The state of each node at a certain time step is linked to its historical states as well its neighbors’ historical states within a time window $Q$. The constructed graph is in a sparse form and thus efficient for computation. We implement the message traverse layer by customizing the HeterGraphConv module of DGL. The code is publicly available at https://github.com/nzhan/TraverseNet.

### IV. EXPERIMENTAL STUDIES

#### A. Dataset

We follow the experimental setup in [32]. We use three traffic datasets, the Caltrans Performance Measurement Sys-
Fig. 4. Cross-correlations between pairs of connected nodes and between pairs of far-away nodes. The x-axis is the time lag. The y-axis is the mean of correlation coefficients with standard deviation. Red lines denote cross-correlations between pairs of connected nodes. Blue lines denote cross-correlations between pairs of far-away nodes. (a) PEMS-04. (b) PEMS-08.

Fig. 5. Distribution of peak points of cross correlation plots between pairs of connected nodes. The x-axis is the time lag. The y-axis is the proportion. (a) PEMS-04. (b) PEMS-08.

| Datasets | # Sensors | Sampling rate | # Time steps | Signals |
|----------|-----------|---------------|--------------|---------|
| PEMS03   | 358       | 5 mins        | 26209        | F       |
| PEMS04   | 307       | 5 mins        | 16992        | F.S.O   |
| PEMS08   | 170       | 5 mins        | 17856        | F.S.O   |

In column titled “Signals”, F represents traffic flow, S represents traffic speed, and O represents traffic occupancy rate.

We provide summary statistics of each dataset in Table II. Two tasks, that is, traffic flow prediction and traffic speed prediction, are evaluated using these datasets. We predict the next 12 steps of traffic speed/flow given the previous 12 steps of traffic signals and the traffic graph. Note that in traffic forecasting, predicting a sequence length of 12 is a long-term prediction as 12 steps already represent a 1-h period. The variation of traffic conditions is very large beyond 1 h. We construct the traffic graph by regarding each sensor as a node and connecting two sensors if they are on the same road. For data preprocessing, we standardize the input to have zero mean and unit variance by

$$\bar{X} = \frac{X - \text{mean}(X)}{\text{std}(X)}.$$  \hfill (12)

To check if the datasets exhibit spatial–temporal dependencies, we plot time lags vs. cross-correlations of pairs of connected nodes and pairs of far-away nodes (i.e., more than nine hops away) for each dataset, respectively. The cross correlation between a sequence \{x_1, x_2, \ldots, x_L\} and a sequence \{y_1, y_2, \ldots, y_L\} at time lag \(k\) is essentially the correlation between the sequence \(y\) and the sequence \(x\) shifted \(k\) steps back

$$C = \frac{\sum_{t=k}^{L} x_{t-k} y_t - \frac{1}{L} \sum_{t=k}^{L} x_{t-k} \sum_{t=k}^{L} y_t}{\sqrt{\frac{1}{L} \sum_{t=k}^{L} x_{t-k}^2 - \left(\frac{1}{L} \sum_{t=k}^{L} x_{t-k}\right)^2} \sqrt{\frac{1}{L} \sum_{t=k}^{L} y_t^2 - \left(\frac{1}{L} \sum_{t=k}^{L} y_t\right)^2}}.$$ \hfill (13)

Fig. 4 and 5 present our analysis. Fig. 4(a) and (b) shows that the cross-correlations between pairs of connected nodes are always higher than those between pairs of far-away nodes across all time lags and datasets. Digging into detail, we plot...
the distribution of peak points of cross-correlation curves between pairs of connected nodes for each dataset, as shown in Fig. 5(a) and (b). The majority of two connected nodes’ cross-correlations peak at time lags 0 and 1. However, there is still a small number of nodes in which the cross-correlation values peak at higher time lags. In particular, for PEMS-04 and PEMS-08, there are 5% and 10% of connected nodes of which the cross-correlation peak at time 11. This suggests that it is reasonable to consider spatial–temporal dependencies explicitly.

B. Baseline Methods

Seven baseline methods are selected in our experiments. Except for STSGCN and STFGNN, we implement all baseline methods in a unified framework. As it is difficult to merge STSGCN and STFGNN into our framework, we directly use the codes of STSGCN and STFGNN in experiments. We give a short description of each baseline method in the following:

1) GRU a sequence-to-sequence model [40] consists of GRU units [41], not considering spatial dependency.
2) Diffusion Convolutional Recurrent Neural Network (DCRNN) [2] that adopts LSTMs and diffusion graph convolution in a sequence-to-sequence framework.
3) Spatio-Temporal Graph Convolutional Network (STGCN) [5] that combines gated temporal convolution with graph convolution to capture spatial dependencies and temporal dependencies, respectively.
4) ASTGCN [8] that interleaves spatial attention with temporal attention to capture dynamic spatial dependencies and temporal dependencies.
5) Graph WaveNet [28] that integrates WaveNet with graph convolution.
6) STSGCN [32] that considers spatial–temporal dependencies in local adjacent spatial–temporal time steps.
7) STFGNN [33] that considers predefined spatial dependencies, precomputed time-series similarities, and local temporal dependencies.

C. Experimental Setting

We conduct the experiments on the Amazon Web Services (AWS) cloud with the p3.8xlarge instance. We train the proposed TraverseNet with the Adam optimizer on a single 16 GB Tesla V100 Graphic Processing Unit (GPU). We split the datasets into train, validation, and test data with a ratio of 6:2:2. We set the number of training epochs to 50, the learning rate to 0.001, the weight decay rate to 0.00001, and the dropout rate to 0.1. We set the number of layers to 3, the hidden feature dimension to 64, and the window size $Q$ to 12. For other baseline methods, we use the default parameter settings reported in their articles. Each experiment is repeated five times and the mean of evaluation metrics including MAE, root mean squared error (RMSE), and mean absolute percentage error (MAPE) on test data are reported based on the best model on the validation data.

D. Overall Results

Table III presents the main experimental results of our TraverseNet compared with baseline methods. There are missing values on PEMS-03 for the traffic speed prediction because PEMS-03 does not contain traffic speed information. Among all methods, TraverseNet achieves the lowest MAE and RMSE on PEMS-03 and PEMS-04 for traffic flow prediction and PEMS-04 for traffic speed prediction. It achieves the second-lowest MAE, MAPE, and RMSE on PEMS-08 for both traffic flow prediction and traffic speed prediction although the performance gap between the top two methods is extremely small.

We believe that the reason that Graph WaveNet performs slightly better than TraverseNet on PEMS-08 is that the spatial signal on that dataset is weak. This is supported by the results of the ablation study (cf. Table IV), which shows that the performance of TraverseNet decreases only slightly when the spatial component of the model is removed. For temporal patterns, the WaveNet component in Graph WaveNet is a very powerful feature extractor for time-series data and this is why its performance is somewhat better than TraverseNet. More importantly, TraverseNet significantly outperforms STSGCN and STFGNN. These two methods have the same motivation as our article which is to jointly handle spatial–temporal dependencies. TraverseNet differs from STSGCN and STFGNN in that they only consider local adjacent spatial–temporal dependencies like RNN-based methods [as demonstrated by Fig. 1(a)], while our method could traverse long-range historical information of a node to another node directly [as demonstrated by Fig. 1(d)].

E. Ablation Study

We perform an ablation study to validate the effectiveness of the message traverse layer in our model. We are mainly concerned about whether the spatial–temporal message traverse layer is effective and whether the attention mechanism in the message traverse layer is useful. To answer these questions, we compare our TraverseNet model with five different settings listed below:

1) **w/o Spatial information**: We only use temporal information, which means the neighborhood set of a node is empty.
2) **w/o st Traversing (without spatial–temporal traversing)**: We handle spatial dependencies and temporal dependencies separately by interleaving spatial attention with temporal attention.
3) **w/o Attention**: We replace attention scores produced by the attention functions with identical weights.
4) **w/o Residual**: We cancel residual connections for message traversing layers and MLP layers.
5) **w/o Norm**: We remove batch normalization after message traversing layers and MLP layers.

We repeat each experiment five times. Table IV reports the mean MAE, MAPE, and RMSE with standard deviation on PEMS-08 test data. We observe that the involvement of spatial information incrementally contributes to model performance. More importantly, spatial–temporal traversing is superior to processing spatial dependencies and temporal dependencies separately based on the fact the performance of **w/o st traversing** is lower than the performance of default. W/o attention nearly does not improve model performance, suggesting that
TABLE III
Performance Comparison

| Task         | Models | PEMS-03 MAE | PEMS-04 MAE | PEMS-08 MAE |
|--------------|--------|-------------|-------------|-------------|
| Traffic Flow | GRU    | 20.01 ± 0.02| 19.82 ± 0.06| 18.86 ± 0.05|
|              | DCRNN  | 16.37 ± 0.05| 15.96 ± 0.10| 17.82 ± 0.13|
|              | STGCN  | 19.58 ± 0.15| 20.17 ± 0.36| 17.40 ± 0.58|
|              | Graph WaveNet | 18.19 ± 0.22| 18.47 ± 0.54| 18.74 ± 0.41|
|              | STP-GCN | 16.74 ± 0.05| 18.56 ± 1.66| 18.75 ± 0.13|
|              | TraverseNet | 17.77 ± 0.20| 17.38 ± 0.08| 17.92 ± 0.14|
|              | 16.56 ± 0.32| 16.09 ± 0.16| 17.75 ± 0.15|
| Traffic Speed| GRU    | -           | -           | -           |
|              | DCRNN  | -           | -           | -           |
|              | STGCN  | -           | -           | -           |
|              | Graph WaveNet | -           | -           | -           |
|              | STP-GCN | -           | -           | -           |
|              | TraverseNet | -           | -           | -           |

The results were obtained on the PEKS-08 dataset.

Fig. 6. MAE plots for hyperparameter study. The y-axis is the MAE score. The mean with one standard deviation is reported as the value of a hyperparameter is increased. (a) Number of layers. (b) Window size. (c) Hidden dimension. (d) Dropout rate. (e) Weight decay. (f) Learning rate.

the effect of attention mechanisms is limited in time-series forecasting. Besides, according to Table IV, the effectiveness of residual connections and batch normalization is verified.

F. Hyperparameter Study
To get an understanding of the effect of key hyperparameters in TraverseNet, we study the effect of varying one hyperparameter at a time while keeping others the same as Section IV-C except that the default number of layers is set to 1 on the validation set of PEKS-08. We vary the number of layers ranging from 1 to 6 by 1, the hidden feature dimension ranging from 32 to 192 by 32, the window size ranging from 2 to 12 by 2, the dropout rate ranging from 0 to 0.5 by 0.1, and both the learning rate and weight decay among \(1 \times 10^{-6}\), \(1 \times 10^{-5}\), \(1 \times 10^{-4}\), \(1 \times 10^{-3}\), \(1 \times 10^{-2}\), and \(1 \times 10^{-1}\). We run each experiment five times. The mean MAE with one standard deviation is reported as the value of a hyperparameter is increased.

TABLE IV
Ablation Study

| PEKMS-08 | w/o spatial info | w/o st traversing | w/o attention | w/o residual | w/o norm | default |
|----------|-----------------|-------------------|--------------|--------------|----------|---------|
| MAE      | 15.95 ± 0.12    | 15.84 ± 0.07      | 15.69 ± 0.11 | 17.01 ± 0.02 | 15.76 ± 0.10 | 15.68 ± 0.12 |
| MAPE     | 10.83 ± 0.20    | 11.05 ± 0.27      | 10.56 ± 0.13 | 13.31 ± 0.90 | 10.86 ± 0.19 | 10.87 ± 0.05 |
| RMSE     | 24.99 ± 0.12    | 24.80 ± 0.07      | 24.66 ± 0.15 | 26.24 ± 0.26 | 24.82 ± 0.10 | 24.62 ± 0.13 |

The mean with one standard deviation is reported as the value of a hyperparameter is increased.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
field of a node, thus a node can track longer and broader neighborhood history. The model performance is not sensitive to the change of hidden dimension. We think it may be due to the nature of time-series data that the input dimension is thin, so a small hidden feature dimension is enough to capture original information. Dropout and weight decay are not necessary for our model since the model performance drops evidently when it increases. Besides, we observe the optimal learning rate is 0.01.

**G. Case Study**

We perform a case study to understand the effect of TraverseNet in capturing inner spatial–temporal dependency. Fig. 7 plots the time series of two neighboring nodes, node 15 and node 151 in PEMS-08. The blue line is the time series of the source node 15 and the yellow line is the time series of the target node 151. We observe that the trend of node 151 follows the trend of node 15 with some extent of latency. For example, the blue line of node 15 starts to drop sharply at step 2, while this phenomenon happens on the yellow line of node 151 6 steps later. Fig. 8 shows that it is not always the case that the cross correlation between the time series of two neighboring nodes is the highest at time step 0. In fact, the time series of node 151 is mostly correlated with the time series of node 15 shifted 6 time steps. Fig. 9 provides a heatmap that visualizes the attention scores produced by TraverseNet in the first message traverse layer between these two time series from time step 0 to time 12. It shows that the state of node 15 at time step 0, 1, 2, and 8 is very important to the state of node 151 at time step 8, 9, and 10. This is consistent with the fact the trend of node 15 at time steps 6, 7, and 8 is similar to the trend of node 151 at time steps 8, 9, and 10 from Fig. 7.

**H. Computation Time**

We compare the computation time of our method with baseline methods on PEMS04 and PEMS08 data in Table V. The training speed of DCRNN is the slowest while the training speed of STGCN is the fastest. The running speed of our method stays in the middle. Although the time complexity of our message traversing layer is $O(M \times p \times Q)$, the speed of our model is still affordable due to an efficient sparse implementation empowered by DGL.

**V. CONCLUSION**

In this article, we propose TraverseNet, a GNN that unifies space and time. The proposed TraverseNet processes a spatial–temporal graph as an inseparable whole. Through our novel message traverse layers, information can be delivered from the neighbors past to the node’s present directly. This design shortens the path length of message passing and enables a node to be aware of its neighborhood variation at firsthand. Experimental results validate the effectiveness of
our framework. As the graph structure is a determinant of a spatial–temporal forecasting model [29], we will consider improving the efficiency of the message traversing by refining the underlying explicit edge relationships in the future.

REFERENCES

[1] Y. Seo, M. Defferrard, P. Vandergheynst, and X. Bresson, “Structured sequence modeling with graph convolutional recurrent networks,” in Proc. Adv. Neural Inf. Process. Syst. Cham, Switzerland: Springer, 2018, pp. 362–373.

[2] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” in Proc. Int. Conf. Learn. Represent., 2018.

[3] Z. Wu, S. Pan, G. Long, X. Wang, and J. Zang, “Graph WaveNet for deep spatial-temporal graph modeling,” in Proc. 28th Int. Joint Conf. Artif. Intell., Aug. 2019, pp. 1907–1913.

[4] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, “Connecting the dots: Multivariate time series forecasting with graph neural networks,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2020, pp. 753–763.

[5] Z. Pan, Y. Liang, W. Wang, Y. Yu, Y. Zheng, and J. Zhang, “Urban traffic prediction from spatio-temporal data using deep meta learning,” in Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Jul. 2019, pp. 1720–1730.

[6] B. Li, X. Li, Z. Zhang, and F. Wu, “Spatio-temporal graph routing for skeleton-based action recognition,” in Proc. AAAI Conf. Artif. Intell., 2019, pp. 8561–8568.

[7] Z. Wu, S. Pan, G. Long, J. Jiang, and C. Zhang, “Spatio-temporal graph structure learning for forecasting,” in Proc. AAAI Conf. Artif. Intell., 2020, vol. 34, no. 1, pp. 1177–1185.

[8] Z. Zhang, X. Fan, C. Wang, and J. Qi, “GMAN: A graph multi-attention network for traffic prediction,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 1234–1241.

[9] C. Song, Y. Lin, S. Guo, and H. Wan, “Spatio-temporal synchronous graph convolutional networks: A new framework for spatio-temporal network data forecasting,” in Proc. AAAI Conf. Artif. Intell., 2020, vol. 34, no. 1, pp. 914–921.

[10] L. Mengzhang and Z. Zhanxing, “Spatio-temporal fusion graph neural networks for traffic flow forecasting,” in Proc. AAAI Conf. Artif. Intell., 2021, pp. 4189–4196.

[11] C. Song, Y. Lin, S. Guo, and H. Wan, “Attention based spatial-temporal graph convolutional networks for traffic flow forecasting,” in Proc. AAAI Conf. Artif. Intell., vol. 33, 2019, pp. 922–929.

[12] C. Park et al., “STGRAT: A spatio-temporal graph attention network for traffic forecasting,” in Proc. Conf. Inf. Knowl. Manage., 2020, pp. 1–8.

[13] J. Zhang, X. Shi, J. Xie, H. Ma, I. King, and D.-Y. Yeung, “GaAN: Gated attention networks for learning on large and spatiotemporal graphs,” in Proc. Conf. Uncertainty Artif. Intell., 2018.

[14] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” in Proc. Int. Conf. Learn. Represent., 2018.

[15] J. Klicpera, S. Weißenberger, and S. Günnemann, “Diffusion improves graph learning,” in Proc. Adv. Neural Inf. Process. Syst., 2019, pp. 13354–13366.

Zonghan Wu received the B.S. degree in systems science from the University of Shanghai for Science and Technology, Shanghai, China, in 2014, and the M.S. degree in statistics from Linköping University, Linköping, Sweden, in 2016. He is currently pursuing the Ph.D. degree in computer science from the University of Technology Sydney (UTS), Ultimo, NSW, Australia. His research concentrates on data mining, machine learning, and deep learning on graphs.
Da Zheng received the B.Sc. degree from Zhejiang University, Hangzhou, China, in 2006, the M.Sc. degree from Ecole Polytechnique fédérale de Lausanne, Lausanne, Switzerland, in 2009, and the Ph.D. degree in computer science from the Johns Hopkins University, Baltimore, MD, USA, in 2017. He is a Senior Applied Scientist with Amazon Web Services Artificial Intelligence (AWS AI), where he develops deep learning frameworks including Mixnet (MXNet), deep graph library (DGL), and DGL-knowledge graph embeddings (KE). His research interest includes high-performance computing, scalable machine learning systems, and data mining.

Shirui Pan received the Ph.D. degree in computer science from the University of Technology Sydney (UTS), Ultimo, NSW, Australia, in 2015. He is an ARC Future Fellow with Monash University, Clayton VIC, Australia. From August 2022, he will be a Professor with Griffith University, Southport, QLD, Australia. His research interests include data mining and machine learning. To date, he has published extensively in top-tier journals and conferences, including the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE (TPAMI), the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS (TNNLS), the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING (TKDE), the International Conference on Machine Learning (ICML), and the SIGKDD Conference on Knowledge Discovery and Data Mining (KDD).

Quan Gan received the bachelor’s degree in computer science from Fudan University, Shanghai, China, in 2016, and the master’s degree in data science from New York University, New York, NY, USA, in 2018. He is currently an Applied Scientist with Amazon Web Services, Seattle, WA, USA.

Guodong Long was born in China. He received the Ph.D. degree in computer science from the University of Technology Sydney, Ultimo, NSW, Australia, in 2014. He is currently an Associate Professor and Core Member with the Center for Artificial Intelligence (CAI), Faculty of Engineering and Information Technology, University of Technology Sydney. His research focuses on machine learning, data mining, and cloud computing.

George Karypis (Fellow, IEEE) is a Distinguished McKnight University Professor and an ADC Chair of Digital Technology with the Department of Computer Science and Engineering, University of Minnesota, Minneapolis, MN, USA. His research interests span the areas of data mining, high-performance computing, information retrieval, collaborative filtering, bioinformatics, cheminformatics, and scientific computing. He has coauthored over 260 articles on these topics and two books (“Introduction to Protein Structure Prediction: Methods and Algorithms” (Wiley, 2010) and “Introduction to Parallel Computing” (Publ. Addison Wesley, 2003, 2nd edition)). He is serving on the program committees of many conferences and workshops on these topics, and on the editorial boards of the IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, Association for Computing Machinery (ACM) Transactions on Knowledge Discovery from Data, Data Mining and Knowledge Discovery, Social Network Analysis and Data Mining Journal, International Journal of Data Mining and Bioinformatics, the journal on Current Proteomics, Advances in Bioinformatics, and Biomedicine and Biotechnology.