# Graph-based Multi-ODE Neural Networks for Spatio-Temporal Traffic Forecasting

**Anonymous authors**
**Paper under double-blind review**

## Abstract

There is a recent surge in the development of spatio-temporal forecasting models in many applications, and traffic forecasting is one of the most important ones. Long-range traffic forecasting, however, remains a challenging task due to the intricate and extensive spatio-temporal correlations observed in traffic networks. Current works primarily rely on road networks with graph structures and learn representations using graph neural networks (GNNs), but this approach suffers from over-smoothing problem in deep architectures. To tackle this problem, recent methods introduced the combination of GNNs with residual connections or neural ordinary differential equations (NODEs). The existing graph ODE models are still limited in feature extraction due to (1) having bias towards global temporal patterns and ignoring local patterns which are crucial in case of unexpected events; (2) missing dynamic semantic edges in the model architecture; and (3) using simple aggregation layers that disregard the high-dimensional feature correlations. In this paper, we propose a novel architecture called Graph-based Multi-ODE Neural Networks (GRAM-ODE) which is designed with multiple connective ODE-GNN modules to learn better representations by capturing different views of complex local and global dynamic spatio-temporal dependencies. We also add some techniques to further improve the communication between different ODE-GNN modules towards the forecasting task. Extensive experiments conducted on four real-world datasets demonstrate the outperformance of GRAM-ODE compared with state-of-the-art baselines as well as the contribution of different GRAM-ODE components to the performance.

## 1 Introduction

Spatio-temporal forecasting is one of the main research topics studied in the context of temporally varying spatial data that arises in many real-world applications such as traffic networks, climate networks, urban systems, etc. In this paper, we study one of the most important applications of spatio-temporal forecasting, traffic forecasting, in which we statistically model and identify historical traffic patterns in conjunction with the underlying road networks to predict the future traffic flow. This task is challenging primarily due to the intricate and extensive spatio-temporal dependencies in traffic networks, also known as intra-dependencies (i.e., temporal correlations within one traffic series) and inter-dependencies (i.e., spatial correlations among correlated traffic series). In addition to this, frequent events such as traffic peaks or accidents lead to the formation of non-stationary time-series among different locations, thus, posing challenges for the prediction task.

Traffic forecasting is a spatio-temporal prediction task that exploits both the location and event data collected by sensors. Traditional methods such as AutoRegressive Integrated Moving Average (ARIMA) and Support Vector Machine (SVM) algorithms only consider the temporal patterns and ignore the corresponding spatial relations (Jeong et al., 2013; Williams & Hoel, 2003; Van Der Voort et al., 1996). Statistical and classical machine learning methods suffer from limitations in learning complex spatio-temporal interactions, and thus deep learning models were later introduced for this task. Early examples of deep learning approaches to capture spatial and temporal relations are FC-LSTM (SHI et al., 2015) where authors integrate CNN and LSTM modules; and ST-ResNet (Zhang et al., 2017) which uses deep residual CNNs for both spatial

and temporal views. However, these CNN-based methods are developed towards grid data and cannnot account for traffic road networks which are more akin to graph-structures. Hence, researchers in the domain have recently developed Graph neural network (GNN) based approaches for effectively learning graph-based representations. Examples of these graph-based models are STGCN (Yu et al., 2018) which utilizes complete convolutional structures for both temporal and spatial views; and DCRNN (Li et al., 2018b) which combines the diffusion process with bi-directional random walks on directed graphs in order to capture spatio-temporal dependencies.

However, such GNN-based methods cannot capture long-range spatio-temporal relations or develop deeper representations due to their limitation of over-smoothing (Li et al., 2018a; Zhou et al., 2020). Deep GNN over-smoothing occurs when a GNN model with deeper architecture tends to lose the discriminative ability and learn similar node representations for all nodes. , thus making it challenging to learn richer representations and investigate more complex spatial features. In order to address this problem, researchers have introduced combining GNNs with residual connections (Chen et al., 2020). Recently, this problem was studied by the STGODE (Fang et al., 2021b) model that integrates GNN-based model with the Neural Ordinary Differential Equations (NODE) (Chen et al., 2018) which can derive a continuous GNN (CGNN) with residual connections, consequently, alleviating the over-smoothing problem. Nevertheless, current CGNN models still encounter the following limitations:

1. Previous approaches in this area tend to overemphasize the global temporal structures while undervaluing local patterns, which are often crucial for predictions in the presence of unexpected traffic events, e.g., a road has a higher chance of getting clogged shortly after a car accident. This will cause drivers to switch to a faster route and thus the traffic flow on this road may significantly drop for a short period of time. Therefore, ignoring these local temporal patterns may cause significant problems in the final predictions for roads facing unexpected events.

2. The dynamic correlation of traffic nodes is ignored by existing approaches. In other words, models usually do not consider the dynamic semantic spatial correlations (i.e., dynamic semantic edges) in their architecture.

3. Several baselines use vanilla aggregations, such as average pooling, to combine latent features learned from multiple streams which ignore the high-dimensional feature interactions.

To overcome the above limitations, we propose a novel framework called **GRAM-ODE**, **GRA**ph-based **M**ulti-**ODE** Neural Networks. First, in order to balance the consideration of global and local temporal patterns, we design new ODE modules for the local temporal patterns in addition to the existing ODE module for the global temporal patterns (Fang et al., 2021b) using different sizes of temporal kernels (represented as grey and orange dashed lines in Fig. 1). Specifically, for local dependencies, we assign ODE functions to the local kernel's output that approximate local patterns, and then concatenate the results. These local and global ODE modules are explained in more detail in Fig. 3(a) and Fig. 3(b), respectively. Second, we design a new ODE module into our model to consider the dynamic correlation of traffic nodes as well as edges. In other words, at each time step, we find the intermediate dynamic spatial correlations based on the embedding representations of nodes (i.e., dynamic semantic edges), and then construct a new ODE module to approximate patterns of semantic edge-to-edge correlations over time (represented with different edge colors and patterns over time in Fig. 1). More details regarding this dynamic semantic edge ODE module are provided in Fig. 3(c). Third, we design the nonlinear aggregation paradigm and a multi-head attention across different ODE modules (represented in Fig. 3(d) ) and different streams of traffic graphs (represented in the last layer of Fig. 2), respectively. Through these two operations, we account for high-dimensional correlations and similarities of latent features corresponding to different ODE modules and traffic graphs. By doing so, we let the model intelligently select and fuse latent features from different views for the intended forecasting task.

Also, since our proposed GRAM-ODE includes multiple ODE modules, we ensure that these different modules are not isolated and have effective connectivity in the intermediate layers. To do so, we design coupled ODE modules in two ways: (1) adding similarity constraint between the semantic parts of local and global modules to ensure these two semantic embeddings do not diverge from one another (represented with orange marks

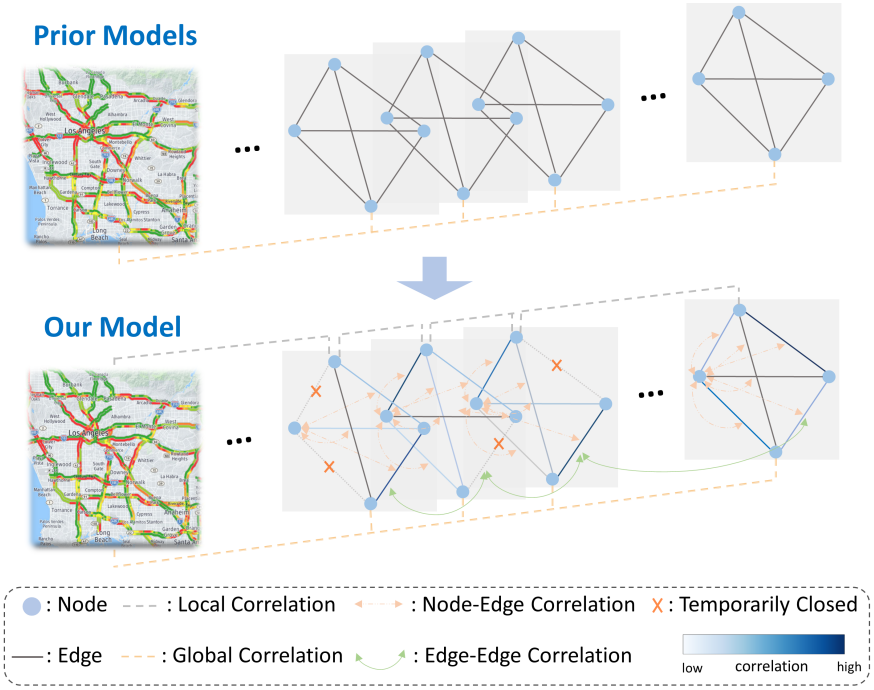


Figure 1: An overview of the proposed model compared to the prior models. The sub-figures from left to right represent the traffic data over time with blue nodes displaying the recording points and the lines between them representing roads. The orange dashed line refers to the common global temporal pattern. In our model, grey dashed line shows the local temporal patterns; orange arrows depict node-edge correlations; and green arcs display dynamic edge-edge correlations. The red cross on some edges implies that the road may be temporarily closed; and the connections with different colors show the dynamic temporal correlations between nodes.

in Fig. 3); (2) sharing weights for the global node-based and edge-based ODE modules (represented with green marks in Fig. 3). Therefore, we create a coupled graph multi-ODE structure as GRAM-ODE in which all modules are designed to effectively communicate with each other for downstream application of traffic prediction. The major contributions of this work are summarized below.

- *Developing a new ODE module for capturing local temporal patterns.* Due to the importance of short-term temporal dependencies in the traffic prediction in case of unexpected events, we develop a new ODE module for the short-term dependencies in addition to the current ODE block for the global temporal patterns. Different sizes of temporal kernels are set to capture different levels of temporal patterns.

- *Developing a new ODE module for the dynamic semantic edges.* In addition to ODE blocks for traffic nodes, which model the dynamic node-to-node correlations over time, we also add a new ODE module for the dynamic semantic edges based on node representations (i.e., dynamic spatial correlations) to model the edge-to-edge correlations over time.

- *Building effective communications between different ODE modules (coupled multi-ODE blocks).* To build effective interactions between multiple ODE modules, we consider shared weights for node-based and edge-based ODE modules as well as adaptive similarity constraints for the outputs of local and global temporal ODE modules.

- *Designing a new aggregation module with multi-head attention across features of different streams.* To enhance the model's awareness in the selection and fusion of different ODE modules as well as the streams corresponding to different types of traffic graphs, we design multi-head attention mechanism at the aggregation layers.

The rest of this paper is organized as follows. Section 2 summarizes existing traffic forecasting works based on machine learning, graph-based and NODE-based methods. Section 3 provides some of the required preliminaries and explains the problem formulation. Section 4 covers the details of our proposed GRAM-ODE method and its different components. Our experimental evaluation including both quantitative and qualitative comparison results, ablation studies, and robustness assessments are reported in Section 5. Finally, Section 6 concludes the paper.

## 2 Related Works

In this section, we describe the traffic forecasting approaches proposed in the literature based on various techniques including machine learning, graph-based methods, and neural differential equations.

### 2.1 Machine Learning and Deep Learning Methods

Researchers have employed traditional statistical and machine learning methods for the task of traffic forecasting. Some prominent example models are (1) K-Nearest Neighbor (KNN) (Zhang et al., 2013) which will predict traffic of a node based on its k-nearest neighbors; (2) ARIMA (Van Der Voort et al., 1996; Alghamdi et al., 2019) which integrates the autoregressive model with moving average operation; and (3) SARIMA (Williams & Hoel, 2003) which adds a specific ability to ARIMA for the recognition of seasonal patterns. Many of these machine learning models only consider the temporal dependencies and ignore the spatial information. Also, they are usually based on human-designed features and have limitations in learning informative features for the intended task. Later, deep learning methods became popular due to their ability in considering the complex and high-dimensional spatio-temporal dependencies through richer representations. Early examples of deep learning models considering the spatial and temporal relations are FC-LSTM (SHI et al., 2015), ConvLSTM (Xingjian et al., 2015), ST-ResNet (Zhang et al., 2017), ST-LSTM (Liu et al., 2016), and STCNN (He et al., 2019) which are usually based on convolutional neural networks (CNNs) and recurrent neural networks (RNNs) to account for spatial and temporal information. However, these models are developed for the grid-based data and disregard the graph structure of traffic road networks. Due to this limitation, researchers moved into the application of graph-based deep learning models like graph neural networks.

### 2.2 Graph-based Methods

Graphs provide vital knowledge about the spatial, temporal, and spatio-temporal relationships that can potentially improve performance of the final model. Recently, researchers employed the graph convolution (GCN) (Kipf & Welling, 2017) to model the spatio-temporal interactions. DCRNN (Veeriah et al., 2015) is an early example of it that utilizes diffusion GCN with bi-directional random walks to model the spatial correlations as well as a GRU (Gated Recurrent Unit) based network to model the temporal correlations. GRU is an RNN-based method which is not effective and efficient in modeling long-range temporal dependencies. To address this limitation, works such as STGCN (Yu et al., 2018) and GraphWaveNet (Wu et al., 2021) utilize convolutional operations for both spatial and temporal domains. After the rise of attention-based models in deep learning, researchers realized that these two models still have some limitations in learning spatial and temporal correlations due to the limited capability of convolutional operations in capturing high-dimensional correlations. Therefore, two attention layers are later employed in ASTGCN (Guo et al., 2019) to capture the spatial and temporal correlations. However, these models are limited in capturing local relations and may lose local information due to the sensitivity of representations to the dilation operation. STSGCN (Song et al., 2020) used localized spatio-temporal subgraphs to enhance prior models in terms of capturing the local correlations. However, since the model formulation does not incorporate global information, this model still had limitations when it came to long-term forecasting and dealing with data that included missing entries or noise. In addition to the spatial graphs from predefined road networks, STFGNN (Li & Zhu, 2021) later introduced the use of Dynamic Time Warping (DTW) for data-driven spatial networks which helped the model to learn representations from different data-driven and domain-driven views. STFGNN also utilized a new fusion module to capture spatial and temporal graphs in parallel. However, due to the over-smoothing problem of deep GNNs (which happens when a GNN model learns similar node representations for all nodes

in case of adopting deeper architecture), current GNN-based models incur some difficulties in learning rich spatio-temporal representations with deep layers.

### 2.3 Neural Differential Equation Methods

Neural Differential Equations (NDEs) (Kidger, 2022) provide a new perspective of optimizing neural networks in a continuous manner using differential equations (DEs). In other words, DEs help to create a continuous depth generation mechanism for neural networks, provide a high-capacity function approximation, and offer strong priors on model space. There are a few types of NDEs: (1) neural ordinary differential equation (NODE) such as ODE-based RNN models (Habiba & Pearlmutter, 2020; Lechner & Hasani, 2022) (e.g., ODE-RNN, ODE-GRU, ODE-LSTM) and latent ODE models (Rubanova et al., 2019); (2) neural controlled differential equation (NCDE) (Kidger et al., 2020; Morrill et al., 2021) which is usually used to learn functions for the analysis of irregular time-series data; and (3) neural stochastic differential equation (NSDE) (Kidger et al., 2021b;a) which is usually employed for generative models that can represent complex stochastic dynamics.

More recently, STGODE (Fang et al., 2021a) is proposed in the traffic forecasting domain to address the aforementioned over-smoothing problem of GNN-based models. STGODE provides a new perspective of continuous depth generation by employing Neural Graph ODEs to capture the spatio-temporal dependencies. However, complex spatio-temporal correlations such as the lack of local temporal dependencies and dynamic node-edge communications are not captured by this model. In this paper, we utilize the idea of Neural Graph ODE to generate multiple coupled ODE-GNN modules which take into account the local temporal patterns and dynamic spatio-temporal dependencies, and as a result, can provide better function approximation for the forecasting in the presence of complex spatio-temporal correlations.

## 3 Problem Formulation

This section explains the required preliminaries and definitions for GRAM-ODE, and then defines the main problem statement. All notations used in this paper are summarized in Table 1.

### 3.1 Definitions

*Definition 1: Traffic Graphs.* We consider the traffic network as a graph $\mathcal{G} = (V, E, A)$, where $V$ is the set of nodes, $E$ is the set of edges and $A$ is the adjacency matrix such that $A \in \mathbb{R}^{N \times N}$ if $|V| = N$. In this paper, we use two types of graphs, the connection map graph and the DTW (dynamic time warping) graph. In the connection map graph, the adjacency matrix, denoted by $A^C$, represents the roads and actual connectivity among traffic nodes with binary value to indicate whether they are connected or not.

$$A^C_{i,j} = \begin{cases} 1, & if\ v_i\ and\ v_j\ are\ neighbors \\ 0, & otherwise \end{cases} \tag{1}$$

where $v_i$ and $v_j$ refer to traffic nodes $i$ and $j$ in the graph. In the DTW graph, the adjacency matrix, indicated by $A^{SE}$, is generated from the Dynamic Time Warping (DTW) algorithm (Berndt & Clifford, 1994) which calculates the distance between two time-series corresponding to a pair of nodes.

$$A^{SE}_{i,j} = \begin{cases} 1, DTW\left(X^i, X^j\right) < \epsilon \\ 0, \text{ otherwise} \end{cases} \tag{2}$$

where $X^i$ and $X^j$ refers to the time-series data of nodes $i$ and $j$, respectively; $\epsilon$ also identifies the sparsity ratio of adjacency matrix. Note that DTW is superior compared to other point-wise similarity metrics (such as Euclidean distance) due to its sensitivity to shape and pattern similarities.

*Definition 2: Graph Normalization.* Adjacency matrix $A \in \{A^C, A^{SE}\} \in \mathbb{R}^{N \times N}$ is normalized with $D^{-\frac{1}{2}} A D^{-\frac{1}{2}}$ where $D$ is the degree matrix of $A$. As shown in Eq. (3), the self-loop identity matrix $I$ is incorporated in normalization to avoid the negative eigenvalues.

$$\hat{A} = \alpha \left( I + D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \right) \tag{3}$$

Table 1: Notations used in this paper.

| Notation | Description |
|---|---|
| $\mathcal{G}$ | Traffic graph |
| $V$ | Set of nodes |
| $E$ | Set of edges |
| $A^C$ | Connection adjacency matrix |
| $A^{SE}$ | DTW adjacency matrix |
| $D$ | Degree matrix |
| $\hat{A}$ | Normalized adjacency matrix |
| $f_e/\ f_g/\ f_l$ | Message generation for Edge/Global/Local Neural ODEs |
| $\mathcal{H}(t)$ | ODE function (integrals from 0 to $t$) initialized with $\mathcal{H}(0)$ |
| $\mathcal{H}_e(t)/\mathcal{H}_g(t)/\mathcal{H}_l(t)$ | Edge/Global/Local ODE function features |
| $\mathcal{X}$ | Historical time series data |
| $\mathcal{Y}$ | Future time series data |
| $\hat{\mathcal{Y}}$ | Predicted future time series |
| $L$ | Length of historical time series |
| $L'$ | length of target time series |
| $N$ | Number of nodes, $\lvert V \rvert$ |
| $C$ | Numbers of channels |
| $H$ | Input of multi ODE-GNN |
| $\mathcal{A}$ | Updated adjacency matrix with shared spatial weights |
| $\mathcal{T}_e/\mathcal{T}_g/\mathcal{T}_l$ | Updated Global/Local/Edge temporal shared weights |
| **EM** | Edge message |
| **GM** | Global message |
| **LM** | Local message |
| $\mathcal{H}_{A_i}(t)$ | Features of single embedded time step in Local ODE-GNN |
| $L''$ | Length of the embedded temporal window in Local ODE-GNN |
| $\mathcal{K}(i)$ | Local ODE-GNN's output for the $i$-th embedded temporal step |
| $e$ | Learnable threshold parameter in Message Filtering |
| $p_n$ | Embeddings of $n$-th ODE module in Aggregation Layer |
| $H'$ | Output of multi ODE-GNN's Aggregation Layer |
| $W_r/b_r$ | Weight matrix/bias vector in Update Layer |
| $H''$ | Output of multi ODE-GNN's Update Layer |
| $H^l_{tcn}$ | Hidden states of $l$-th TCN's layer |
| $C^l$ | Embedding size of $l$-th TCN's layer |
| $W^l$ | Convolutional kernel of $l$-th TCN's layer |
| $d^l$ | Exponential dilation rate of $l$-th TCN's layer |
| $W_q/\ W_k/\ W_v$ | The attention Query/Key/Value weight matrix |
| $b_q/\ b_k/\ b_v$ | The attention Query/Key/Value bias vector |
| $h$ | Number of attention heads |
| $X'_i$ | Attention output of the $i$-th head |
| $\delta$ | Error threshold in the Huber Loss |

where $\hat{A}$ is the normalized adjacency matrix; and $\alpha \in (0, 1)$ is a hyper-parameter identifying the eigenvalue range to be in $[0, \alpha]$.

*Definition 3: Graph-based Neural ODE.* The standard formulation of GNN-based continuous-time ODE function is defined in Eq. (4). It takes the initial value $\mathcal{H}(0)$, temporal integrals from 0 to given time $t$, traffic graph $\mathcal{G}$, and the Neural ODE's network parameter $\theta$.

$$\mathcal{H}(t) = \mathcal{H}(0) + \int_0^t f(\mathcal{H}(s), s; \mathcal{G}, \theta)\mathrm{d}s \tag{4}$$

where $f$ is the process to generate semantic message of Neural ODE parameterized by $\theta$ to model the hidden dynamics of graph $\mathcal{G}$. Since the structure of our proposed method consists of multiple ODE-GNN blocks, we have different types of Eq. (4) graph neural ODEs which are elaborated with more details in Section 4.

*Definition 4: Tensor n-mode multiplication.* We use subscript to identify the tensor-matrix multiplication on the specific dimension as shown below.

$$(\mathcal{B} \times_2 \mathcal{C})_{ilk} = \sum_{j=1}^{n_2} \mathcal{B}_{ijk}\mathcal{C}_{jl} \tag{5}$$

where $\mathcal{B} \in \mathbb{R}^{N_1 \times N_2 \times N_3}$, $\mathcal{C} \in \mathbb{R}^{N_2 \times N'_2}$, so, $\mathcal{B} \times_2 \mathcal{C} \in \mathbb{R}^{N_1 \times N'_2 \times N_3}$. The $n$-mode tensor-matrix multiplication is denoted as $\times_n$ with the $n$-th subscript.

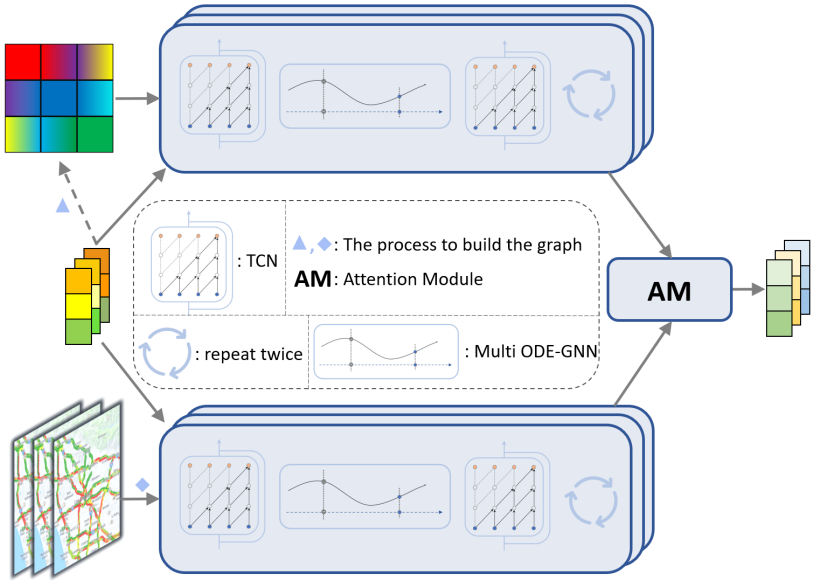


Figure 2: A graphical representation of the proposed GRAM-ODE framework. The DTW-based graph will be obtained from data at the blue triangle mark. The connection map will be obtained from the distance among recording nodes at the blue diamond mark. These two traffic graphs will be separately imported into the model which consists of three parallel channels of two consecutive GRAM-ODE layers. Each layer contains a sandwiched multi ODE-GNN block (check Fig. 3 for more details) between two temporal convolution networks (TCNs). Outputs of these two streams will be aggregated with an Attention Module in the final layer.

### 3.2 Problem Statement

The spatio-temporal forecasting problem is described as learning a mapping function $\mathcal{F}$ that transforms the current historical spatio-temporal data $\mathcal{X} = (X_{t-L+1}, X_{t-L+2}, ..., X_t)$ into the future spatio-temporal data $\mathcal{Y} = (X_{t+1}, X_{t+2}, ..., X_{t+L'})$, where $L$ and $L'$ denote the length of historical and target time-series to be predicted, respectively. In the traffic forecasting problem, we have a historical tensor $\mathcal{X} \in \mathbb{R}^{B \times N \times L \times C}$ and traffic graph $\mathcal{G}$, where $B$ is the batch size; $N$ is the number of nodes; $L$ is the input temporal length; and $C$ is the number of input features (e.g., traffic speed, flow, density). The goal is to find $\hat{\mathcal{Y}} = \mathcal{F}(\mathcal{X}, f, \mathcal{G})$ in which $\mathcal{F}$ is the overall forecasting network and $f$ corresponds to different graph-based Neural ODE processes.

## 4 GRAM-ODE

The general structure of our proposed GRAM-ODE is shown in Fig. 2, which is composed of two streams of operations: $(i)$ DTW-based graph (top row) which is constructed based on semantical similarities, and $(ii)$ connection map graph (bottom row) which is constructed based on geographical spatial connectivities. These two types of adjacency matrices will be fed into the model separately to capture spatial correlations from both data-driven and geographical domain-driven views. They will be later integrated with the multi-head attention mechanism in the final layer. As shown in Fig. 2, we have three parallel channels for each graph, each of which has two consecutive GRAM-ODE layers with a multi ODE-GNN block sandwiched between two blocks of temporal dilated convolution (TCN). The final features of each graph from different channels will then be concatenated and imported into the attention module which is designed to fuse these two separate set of operations more intelligently towards the intended forecasting task. We provide further details about each of these components in the subsections below.

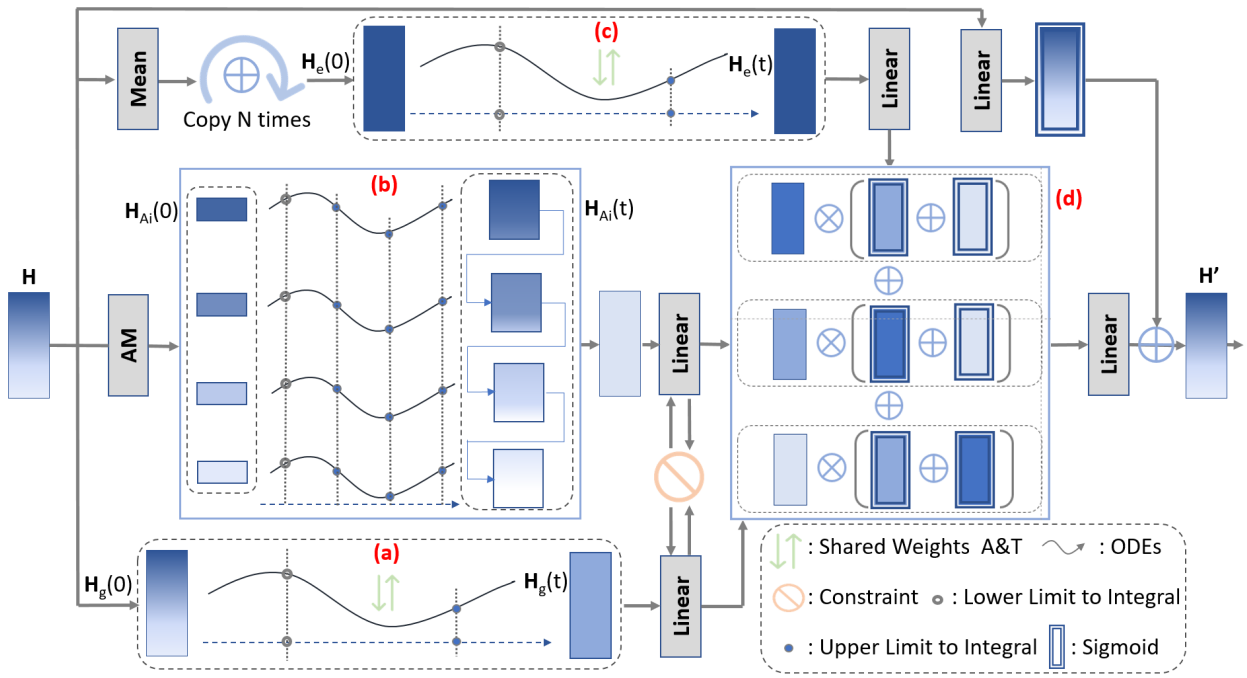


Figure 3: An overview of the multi ODE-GNN block which consists of three ODE modules, i.e., (a) global, (b) local, and (c) edge-based temporal dependencies as well as a new aggregation layer (d). The inputs and outputs of the multi ODE-GNN block are displayed with $H$ and $H'$ blocks on the left and right sides of the diagram. It uses shared weights and message filtering constraint to make the outputs of different ODE modules communicate. These shared weights among different ODE modules are represented with the green mark. The constraint to limit divergence of embeddings is also represented with orange mark.

### 4.1 GRAM-ODE Layer

Each GRAM-ODE layer consists of a multi ODE-GNN block placed between two TCNs. Fig. 3 illustrates details of our proposed Multi ODE-GNN block in which we combine multiple ODE modules to take into account all the essential temporal and spatial patterns from different viewpoints and extract informative features from their fusion. In the multi ODE-GNN block, we design three types of message passing operations, a message filtering constraint as well as a new aggregation module to consider high-dimensional correlations.

#### 4.1.1 Message Passing Layer

The current models primarily focus on node-based global temporal correlations and do not consider short-term temporal patterns as well as the dynamic edge-based correlations into the model formulation. Hence, we design three types of message passing processes to formulate ODE modules corresponding to global, local, and edge-based temporal dependencies.

*Shared Weights:* Despite different semantic meaning corresponding to node-based and edge-based features, they can share some important spatial and temporal patterns. In order to consider these common patterns in our problem formulation, we design a shared weight matrix between node-based and edge-based global temporal ODE modules (shown in Fig. 3(a) and 3(c) ). We define two weight matrices for consideration of these shared spatial weights and shared temporal weights. The shared spatial weight, denoted by $M$, will be added to the normalized adjacency matrix as $\hat{\mathcal{A}} = \hat{A} + M$, where $M$ is initialized randomly from normal distribution. The shared temporal weights, denoted by $W_{s1}, W_{s2}$, will be added into the node-based and edge-based global temporal modules, given by Eqs. (6) and (7), respectively. To consider the local temporal patterns in model formulation, we apply different sizes of temporal kernels $W_{l1}, W_{l2}$ into the local temporal

module as shown in Eq. (8).

$$\mathcal{T}_e = (\mathcal{H}_e(t) \times_4 W_{s1})^T (\mathcal{H}_e(t) \times_4 W_{s2}) \tag{6}$$

$$\mathcal{T}_g = (\mathcal{H}_g^T(t) \times_3 W_{s1})(\mathcal{H}_g^T(t) \times_3 W_{s2})^T \tag{7}$$

$$\mathcal{T}_l = (\mathcal{H}_l^T(t) \times_3 W_{l1})(\mathcal{H}_l^T(t) \times_3 W_{l2})^T \tag{8}$$

where $W_{s1}, W_{s2} \in \mathbb{R}^{L \times L}$ represent the global temporal shared weights with $L$ referring to the global temporal length; $W_{l1}, W_{l2} \in \mathbb{R}^{1 \times 1}$ represent the local temporal weights for each time step in the embedded temporal space; and $\mathcal{H}_e(t), \mathcal{H}_g(t)$, and $\mathcal{H}_l(t)$ represent the feature of edge, global, and local ODE module, respectively.

*Global and Local Message Passing:* Fig 3(a) represents the global message passing with the goal of modeling long-term node-based temporal patterns in a continuous manner. Eqs. (9) and (10) represent the global message processing operations and return the global message **GM**, respectively. Additionally, Fig 3(b) represents the local message passing operations on local temporal data to emphasize the importance of short-term temporal patterns. In this module, we will first use self-attention mechanism with dense projection layers to create the input in the lower-dimensional embedded temporal space by considering all the possible high-dimensional correlations (similar to the attention module ($AM$) explained with more details in Section 4.2). Then, features of each time stamp in the embedded temporal inputs will be separately imported into the local ODE functions, encouraging the network to learn implicit local dependencies. These features at each embedded time step are denoted by $\mathcal{H}_{Ai} \in \mathbb{R}^{B \times N \times 1 \times C}$, where $i \in \{0, 1, 2, 3, ..., L'' - 1\}$, and $L''$ is the size of the embedded temporal window. As shown in Eqs. (13) and (15), at each embedded time step, $\mathcal{H}_{Ai}$ will be used as the initial value to formulate temporal dependencies of future $t = L/L''$ time stamps which will be returned as $\mathcal{K}(i)$. Then, the outputs will be concatenated to form the final local message $\mathbf{LM} = \mathcal{K}(0)||\mathcal{K}(1)||\ldots||\mathcal{K}(L'' - 1)$.

$$\mathbf{GM} = GlobalMessagePassing(\mathcal{H}_g(0), \hat{\mathcal{A}}, \mathcal{T}_g, \mathcal{W}) = \mathcal{H}_g(0) + \int_0^t f_g(\mathcal{H}_g(\tau), \hat{\mathcal{A}}, \mathcal{T}_g, \mathcal{W}) \mathrm{d}\tau \tag{9}$$

$$f_g = \mathcal{H}_g(t) \times_2 (\hat{\mathcal{A}} - I) + ((S(\mathcal{T}_g) - I)\mathcal{H}_g^T(t))^T + \mathcal{H}_g(t) \times_4 (\mathcal{W} - I) \tag{10}$$

$$\mathbf{LM} = LocalMessagePassing(ATT, \mathcal{H}_l(0), \hat{\mathcal{A}}, \mathcal{T}_l, \mathcal{W}) = \mathcal{K}(0)||\mathcal{K}(1)||.....||\mathcal{K}(L'' - 1) \tag{11}$$

$$\mathcal{H}_{A0}, \mathcal{H}_{A1}, ..., \mathcal{H}_{Al} = ATT(\mathcal{H}_l(0)) \tag{12}$$

$$\mathcal{K}(i) = F_l(\mathcal{H}_{Ai}, t_0)||F_l(\mathcal{H}_{Ai}, t_1)||.....||F_l(\mathcal{H}_{Ai}, t_{L/L''-1}) \tag{13}$$

$$F_l(\mathcal{H}_{Ai}, t_j) = \mathcal{H}_{Ai} + \int_0^{t_j} f_l(\mathcal{H}_{Ai}(\tau), \hat{\mathcal{A}}, \mathcal{T}_l, \mathcal{W}) \mathrm{d}\tau \tag{14}$$

$$f_l = \mathcal{H}_{Ai}(t) \times_2 (\hat{\mathcal{A}} - I) + ((S(\mathcal{T}_l) - I)\mathcal{H}_{Ai}^T(t))^T + \mathcal{H}_{Ai}(t) \times_4 (\mathcal{W} - I) \tag{15}$$

*Edge Message Passing:* Fig. 3(c) depicts the edge message passing procedures that take into account the dynamic edge-based temporal patterns in addition to the node-based correlations. We will first create the initial edge features $H_e(0) \in \mathbb{R}^{B \times N \times N \times L}$ from the node representation $H \in \mathbb{R}^{B \times N \times L \times C}$ by taking an average over $C$ channels and copying the $N$ dimension. Eqs. (16) and (17) represent the edge message passing operations which return edge message **EM**.

$$\mathbf{EM} = EdgeMessagePassing(\mathcal{H}_e(0), \hat{\mathcal{A}}, \mathcal{T}_e) = \mathcal{H}_e(0) + \int_0^t f_e(\mathcal{H}_e(\tau), \hat{\mathcal{A}}, \mathcal{T}_e) \mathrm{d}\tau \tag{16}$$

$$f_e = \mathcal{H}_e(t) \times_2 (\hat{\mathcal{A}} - I) + \mathcal{H}_e(t)(S(\mathcal{T}_e) - I) \tag{17}$$

In Eqs (9)-(17), $\mathcal{H}_g$, $\mathcal{H}_l$, and $\mathcal{H}_e$ represent the feature of global, local, and edge ODE modules, repsectively; $\mathcal{T}_g$, $\mathcal{T}_l$, and $\mathcal{T}_e$ represent the global, local, and edge temporal weights formulated in Eqs. (6)-(8), respectively; $\hat{\mathcal{A}}$ represents the updated adjacency matrix based on shared spatial weights; $\mathcal{W} \in \mathbb{R}^{C \times C}$ represents the weight matrix for modeling interaction of different channels; $S(.)$ shows the sigmoid operation; and $I$ is the identical matrix.

#### 4.1.2 Message Filter

To prevent the semantic parts of local and global modules diverging from one another, we add a message filtering constraint into the problem formulation (shown with orange mark in Fig. 3). Eq. (18) shows that this message filtering constraint works like a both-way clipping operation in which we clip local semantic embeddings if they are significantly larger or smaller than global semantic embeddings.

$$LM = \begin{cases} GM + e, \, if \quad LM > GM + e \\ GM - e, \, if \quad LM < GM - e \\ LM, \, otherwise \end{cases} \tag{18}$$

where $LM$ and $GM$ represent the local and global semantic embeddings; and $e$ represents a learnable noise parameter initialized with normal distribution.

#### 4.1.3 Aggregation Layer

Fig. 3(d) represents our aggregation paradigm designed for combining output features of the three ODE modules. Instead of employing single add or pooling operations in the aggregation layer, we use the non-linear matrix multiplication operation to account for key correlations in the higher dimensions. Eq. (19) represents our designed aggregation in which the output of each ODE module is multiplied by the sum of other modules that are normalized by softmax operation. This gated aggregation has several benefits: $(i)$ it enables the selection of features that are more crucial for forecasting; $(ii)$ it allows for non-linear aggregation; and $(iii)$ it contains a linear gradient path, reducing the risk of vanishing gradient.

$$H' = Aggregation(GM, LM, EM) = \frac{1}{2K} \sum_{m}^{K} \sum_{n \neq m}^{K} p_m \odot softmax(p_n) \tag{19}$$

where $H'$ represents the output of Multi ODE-GNN aggregation layer; $LM$, $GM$ and $EM$ represent the local, global, and edge-based semantic embeddings, respectively; $K = 3$ refers to three ODE modules ($p_0 = GM, \; p_1 = LM, \; p_2 = EM$); and $\odot$ operation represents the point-wise multiplication.

#### 4.1.4 Update Layer

After obtaining the aggregated information, we add a residual connection around the multi ODE-GNN block to update the node information of GNN. To achieve this, the input of multi ODE-GNN block will be remapped to the aggregated output using a Fully Connected Layer as shown below.

$$H'' = Update(H', H) = \alpha * Sigmoid(W_r H + b_r) + \beta * H' \tag{20}$$

where $H$ and $H'$ represent the inputs and outputs of the multi ODE-GNN block, respectively; $W_r$ and $b_r$ denote the weight matrix and bias vector of the residual fully connected layer, respectively. Also, $\alpha$ and $\beta$ are the hyperparameters identifying the combination weights in the residual connection.

#### 4.1.5 Temporal Convolutional Network

Since the problem being considered in this paper is spatio-temporal in nature, we need to consider the temporal correlations of nodes in addition to their spatial relations in the problem formulation. To model the temporal correlations, we utilize temporal convolutional networks (TCNs) which usually have more efficient training, more stable gradients, and faster response to dynamic changes compared to recurrent neural networks (RNNs). The architecture of TCN adopts the dilated convolutional operations with 1-D kernels along the time axis.

$$H_{tcn}^{l} = \begin{cases} X & , \quad l = 0 \\ Sigmoid\left(W^{l} \odot d^{l} H_{tcn}^{l-1}\right) & , \quad l = 1, 2, \ldots, L \end{cases} \tag{21}$$

where $X \in \mathbb{R}^{B \times N \times L \times C}$ is the input of TCN; $H_{tcn}^{l} \in \mathbb{R}^{B \times N \times L \times C^{l}}$ is the latent output with $C^{l}$ as the embedding size in the $l$-th TCN's layer; $W^{l}$ represents the convolutional kernel of the $l$-th layer; and $d^{l}$ is the exponential dilation rate which is used to expand the receptive field during the convolution. We usually take $d^{l} = 2^{l-1}$ in the temporal dilated convolution.

Algorithm 1 provides the pseudocode for the GRAM-ODE layer by sequentially passing the input via TCN, Multi ODE-GNN, and another TCN blocks. The previously explained steps of Multi ODE-GNN block including initialization, message passing, message filtering, aggregation, and update are summarized in lines 4 - 26.

**Algorithm 1:** GRAM-ODE Layer

**Input:** Node Information $\mathcal{X}$, Traffic Graph $\mathcal{G}$
**Output:** Updated Node Information $\hat{\mathcal{X}}$

1 # TCN Block
2 $H \leftarrow \text{TCN}(\mathcal{X})$
3 # Find Initial Values
4 $\mathcal{H}_g(0) \leftarrow H$
5 $\mathcal{H}_{Ai}(0) \leftarrow \text{ATT}(H)$
6 $\hat{H} \leftarrow \text{Mean}(H)$ # average on channel dimension
7 $H_e(0) \leftarrow \text{Repeat}(\hat{H})$ # repeat N times
8 # Edge Message Passing
9 $EM \leftarrow \mathcal{H}_e(t) \times_2 (\hat{\mathcal{A}} - I) + \mathcal{H}_e(t)(S(\mathcal{T}_e) - I)$
10 # Global Message Passing
11 $GM \leftarrow \mathcal{H}_g(t) \times_2 (\hat{\mathcal{A}} - I) + ((S(\mathcal{T}_g) - I)\mathcal{H}_g^T(t))^T + \mathcal{H}_g(t) \times_4 (\mathcal{W} - I)$
12 # Local Message Passing
13 $LM \leftarrow \mathcal{H}_{Ai}(t) \times_2 (\hat{\mathcal{A}} - I) + ((S(\mathcal{T}_l) - I)\mathcal{H}_{Ai}^T(t))^T + \mathcal{H}_{Ai}(t) \times_4 (\mathcal{W} - I)$
14 # Message Filtering
15 **if** $LM > GM + e$ **then**
16 | $LM \leftarrow GM + e$
17 **else if** $LM < GM - e$ **then**
18 | $LM \leftarrow GM - e$
19 **else**
20 | $LM \leftarrow LM$
21 **end**
22 # Aggregate Multi ODE-GNNs
23 $p_0, p_1, p_2 \leftarrow GM, LM, EM$
24 $H' \leftarrow \frac{1}{2K} \sum_{m}^{K} \sum_{n \neq m}^{K} p_m \odot softmax(p_n)$
25 # Update
26 $H'' \leftarrow \alpha * Sigmoid(W_r H + b_r) + \beta * H'$
27 # TCN Block
28 $\hat{\mathcal{X}} \leftarrow \text{TCN}(H'')$

### 4.2 Attention Module

We use the attention mechanism in the last layer of GRAM-ODE to aggregate the final learned embeddings of two traffic graphs (i.e., DTW-based graph and the connection map graph) in a more intelligent way towards the forecasting objective. The attention module (AM) is designed to replace the previous fully connected layers while capturing the correlations of high-dimensional information. In this module, we will first concatenate the embeddings of two graphs and then compute the attention scores between them. Eqs. (22) and (23) mathematically describe this attention operation.

$$Q = XW_q + b_q, \; K = XW_k + b_k, \; V = XW_v + b_v \tag{22}$$

$$X'_i = softmax\,(\sqrt{\frac{h}{C'}} * (Q_i^T K_i))\, V_i \tag{23}$$

where $W_q(b_q), W_k(b_k)$, and $W_v(b_v)$ represent the attention query, key, and value weight matrix (bias vector), respectively; $X$ is the input of attention module; $h$ is the number of heads; and $\sqrt{\frac{h}{C'}}$ is the normalization factor with $C'$ representing the embedding dimension. Therefore, $Q$:$\{Q_1, Q_2, ..., Q_h\}$, $K$:$\{K_1, K_2, ..., K_h\}$, $V$:$\{V_1, V_2, ..., V_h\}$ shows the query, key, and value set for multiple attention heads. Finally, output of the attention module $X'_i$ can be mapped to the feature space of original data for prediction with a linear dense layer.

### 4.3 Loss Function

In regression problem settings (such as the traffic flow forecasting), Huber loss function between the real value and predicted value is widely used in the literature. It combines the merits of $L1$ and $L2$ loss functions. Huber loss is a piece-wise function which consists of two parts: (1) a squared term with small and smooth gradient when the difference between the true and predicted values is small (i.e., less than a $\delta$ threshold value) and (2) a restricted gradient term when the true and predicted values are far from each other. Eq. (24) represents the standard form of Huber loss function.

$$L(\hat{\mathcal{Y}}, \mathcal{Y}) = \begin{cases} \frac{1}{2}(\hat{\mathcal{Y}} - \mathcal{Y})^2, & |\hat{\mathcal{Y}} - \mathcal{Y}| \leq \delta \\ \delta|\hat{\mathcal{Y}} - \mathcal{Y}| - \frac{1}{2}\delta^2, & otherwise \end{cases} \tag{24}$$

where $\delta$ is a hyperparameter value set for the intended threshold; $\mathcal{Y}$ is the true future spatio-temporal data; and $\hat{\mathcal{Y}}$ is the predicted future data.

Algorithm 2 provides the pseudocode of GRAM-ODE training. In each optimization step of this algorithm, the outputs of all GRAM-ODE layers across parallel channels will be concatenated, and these concatenated outputs of different graph types will then be fed into the attention module for better aggregation and prediction.

**Algorithm 2:** GRAM-ODE training

**Input:** Historical Data $\mathcal{X}$, Future Data $\mathcal{Y}$, Traffic Graph $\mathcal{G}$
**Output:** Forecast Model with parameter $\theta$

```
initialize model parameters θ
normalize the historical data X ← (X − mean(X)) / std(X)
for number of epochs until convergence do
    for batch in num_batches do
        layers=[ ]
        for graph g in G do
            repeat
                X̂ ← GRAM-ODE Layer(X, g)
                layers ←[layers, (X̂)]  # Concatenation
            until num_paralleled_layers
        end
        X′ ← Attention Module(layers)
        Ŷ ← X′ × std(X) + mean(X)
        l ← Huber Loss(Ŷ, Y) # Compute Loss
        θ ← θ − Δ_θ l # Update Parameters
    end
end
```

## 5 Our Experiments

We conduct experiments on four real-wold datasets and seven baseline models to evaluate the effectiveness of our proposed GRAM-ODE and its components for the traffic forecasting task.

### 5.1 Datasets

We use four widely used public traffic datasets [1] (PEMS03, PEMS04, PEMS07, and PEMS08) which are collected from the Freeway Performance Measurement System in California, USA. All these datasets collect three features (flow, occupation, and speed) at each location point over a period of time (with 5-minute time intervals). The spatial connection network for each dataset is constructed using the existing road network. The details of data statistics are shown in Table. 2. To use these datasets in experiments, we pre-process the features by z-score normalization.

Table 2: Basic statistics of the datasets used in our experiments.

| **Data** | **PEMS03** | **PEMS04** | **PEMS07** | **PEMS08** |
|---|---|---|---|---|
| Location | CA, USA | | | |
| Time Span | 9/1/2018 - 11/30/2018 | 1/1/2018 - 2/28/2018 | 5/1/2017 - 8/31/2017 | 7/1/2016 - 8/31/2016 |
| Time Interval | 5 min | | | |
| Sensors | 358 | 307 | 883 | 170 |
| Edges | 547 | 340 | 866 | 295 |
| Time Steps | 26208 | 16992 | 28224 | 17856 |

### 5.2 Evaluation Metrics

We use Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE) metrics to evaluate the spatio-temporal forecasting. These metrics are defined as follows.

$$\begin{aligned} MAE =& \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \\ MAPE =& (\frac{1}{n}\sum_{i=1}^{n}|\frac{y_i - \hat{y}_i}{y_i}|) * 100\% \\ RMSE =& \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{y}_i\right)^2} \end{aligned} \tag{25}$$

### 5.3 Baselines

We compared our proposed GRAM-ODE with the following baselines.

- **ARIMA**(Box & Pierce, 1970): Auto-Regressive Integrated Moving Average is one of the most well-known statistical models for time-series analysis.
- **DCRNN**(Veeriah et al., 2015): Diffusion Convolutional Recurrent Neural Network utilizes diffusion graph convolutional networks with bidirectional random walks on directed graphs, and seq2seq gated recurrent unit (GRU) to capture spatial and temporal dependencies, respectively.
- **STGCN** (Yan et al., 2018): Spatio-Temporal Graph Convolutional Network combines graph structure convolutions with 1D temporal convolutional kernels to capture spatial dependencies and temporal correlations, respectively.
- **GraphWaveNet** (Wu et al., 2021): Graph WaveNet integrates adaptive graph convolution with 1D dilated casual convolution to capture spatio-temporal dependencies.

[1] All four datasets are provided in STSGCN github repository https://github.com/Davidham3/STSGCN/

Table 3: Performance comparison of GRAM-ODE and baselines on four datasets. A lower MAE/MAPE/RMSE indicates better performance. The **best** results are in bold and the second-best are underlined.

| Dataset | Metric | ARIMA | DCRNN | STGCN | GraphWaveNet | STSGCN | STFGNN | STGODE | **GRAM-ODE** |
|---|---|---|---|---|---|---|---|---|---|
| PEMS03 | MAE | 33.51 | 18.18 | 17.48 | 19.85 | 17.48 | 16.77 | 16.50 | **15.72** |
| | MAPE(%) | 33.78 | 18.91 | 17.15 | 19.31 | 16.78 | 16.30 | 16.69 | **15.98** |
| | RMSE | 47.59 | 30.31 | 30.12 | 32.94 | 29.21 | 28.34 | 27.84 | **26.40** |
| PEMS04 | MAE | 33.73 | 24.70 | 22.70 | 25.45 | 21.19 | 20.84 | 20.84 | **19.55** |
| | MAPE(%) | 24.18 | 17.12 | 14.59 | 17.29 | 13.90 | 13.02 | 13.77 | **12.66** |
| | RMSE | 48.80 | 38.12 | 35.55 | 39.70 | 33.65 | 32.51 | 32.82 | **31.05** |
| PEMS07 | MAE | 38.17 | 25.30 | 25.38 | 26.85 | 24.26 | 23.46 | 22.99 | **21.75** |
| | MAPE(%) | 19.46 | 11.16 | 11.08 | 12.12 | 10.21 | **9.21** | 10.14 | 9.74 |
| | RMSE | 59.27 | 38.58 | 38.78 | 42.78 | 39.03 | 36.60 | 37.54 | **34.42** |
| PEMS08 | MAE | 31.09 | 17.86 | 18.02 | 19.13 | 17.13 | 16.94 | 16.81 | **16.05** |
| | MAPE(%) | 22.73 | 11.45 | 11.40 | 12.68 | 10.96 | 10.60 | 10.62 | **10.58** |
| | RMSE | 44.32 | 27.83 | 27.83 | 31.05 | 26.80 | 26.22 | 25.97 | **25.17** |

- **STSGCN** (Song et al., 2020): Spatio-Temporal Synchronous Graph Convolutional Networks decompose the problem into multiple localized spatio-temporal subgraphs, assisting the network in better capturing of spatio-temporal local correlations and consideration of various heterogeneities in spatio-temporal data.
- **STFGNN** (Li & Zhu, 2021): Spatio-Temporal Fusion Graph Neural Networks uses Dynamic Time Warping (DTW) algorithm to gain features, and follow STSGCN (Song et al., 2020) in using sliding window to capture spatial, temporal, and spatio-temporal dependencies.
- **STGODE** (Fang et al., 2021a): Spatio-Temporal Graph ODE Networks attempt to bridge continuous differential equations to the node representations of road networks in the area of traffic forecasting.

### 5.4 Experimental Settings

Following the previous works in this domain, we perform 5-fold cross validation experiments by splitting the entire dataset into 3:1:1 for train, validation, and test sets. We use the past 1 hour to predict the future 1 hour. Since the time interval of data collection is 5 minutes, $L, L' = 12$ temporal data points. Based on Table 2, we have different number of sensors among different datasets, therefore, $|V|$ will be different. DTW threshold ($\epsilon$) in Eq. (2) is 0.1; number of channels ($C$) in the historical data is 3 (i.e., flow, speed, and occupation) and in the embedding space is 64. The shared temporal weights $W_{s1}, W_{s2} \in \mathbb{R}^{12\times 12}$ are initialized randomly from normal distribution. The length of latent space for the input of local ODE block is $L'' = 4$, and in the final attention module, number of attention heads $h = 12$. During training, we use the learning rate of $10^{-4}$, $10^{-4}$, $10^{-5}$, and $10^{-5}$ for PEMS03, PEMS04, PEMS07, and PEMS08 datasets, respectively. The optimizer is AdamW. All experiments are implemented using PyTorch (Paszke et al., 2019) and trained using Quadro RTX 8000 GPU, with 48GB of RAM. The source code for our model will be available at `https://anonymous.4open.science/r/GRAM-ODE-FBF6/` upon the acceptance of this paper.

### 5.5 Experimental Results

Table 3 shows that our proposed GRAM-ODE outperforms other baselines on all the datasets with different metrics (with the only exception of MAPE metric in PEMS07 - which is slightly larger than that of STFGNN). In this table, ARIMA performs considerably worse than other baselines which is probably because it ignores the graph structure of spatio-temporal data. GraphWaveNet is also performing comparatively poor which might be due to the limited capability of this method in stacking spatio-temporal layers and expanding the receptive field learned from 1D CNN temporal kernels. DCRNN uses bi-directional random walks to model spatial information, while a gated recurrent unit (GRU) is used to model temporal information.

The GRU is an RNN-based operation with a relatively low modeling efficacy and efficiency for long-range temporal information, which may contribute to its subpar performance. STGCN uses graph convolutional network (GCN) for spatial domain and 1D dilated convolutional operations for the temporal domain which is more efficient than RNN-based operations but may cause some performance issues, such as losing the local information because of the dilation operation. The relatively poor performance of STGCN might also be due to the absence of attention-based operations and the limited capability of convolutional operations in the modeling high-dimensional spatial, temporal, and spatio-temporal correlations. STSGCN employs the localized spatio-temporal subgraphs to capture the local correlations, however this method missed the global information, potentially resulting in poor performance in the case of long-range forecasting and data with missing entries or noise. In addition to the spatial graphs from road networks, STFGNN uses Dynamic Time Warping (DTW) for latent spatial networks which offered considerably good performance. However, STFGNN is still limited in the learning of comprehensive spatio-temporal correlations. STGODE uses Neural ODEs to capture the spatio-temporal dynamics, resulted in a very good performance compared to other baselines. However, STGODE is still not very strong as it is unable to capture complicated spatio-temporal correlations, balance local and global patterns, or capture the dynamic interactions between its various components.

### 5.6 Case Study

We selected two nodes from the PEMS08 road networks to conduct our case study for the qualitative demonstration of results. As Fig.4 shows, the predicted curves by our proposed GRAM-ODE (red curve) is better (more closely) aligned with the ground truth than STGODE (grey curve). The ground truth in node 109 has more fluctuations compared to the node 17 which causes more difficulty in the forecasting task. We can also observe that our model provides faster response in case of these abrupt fluctuations. This highlights the effectiveness of local ODE-GNN block in our model architecture which helps the model to better learn the local fluctuations.

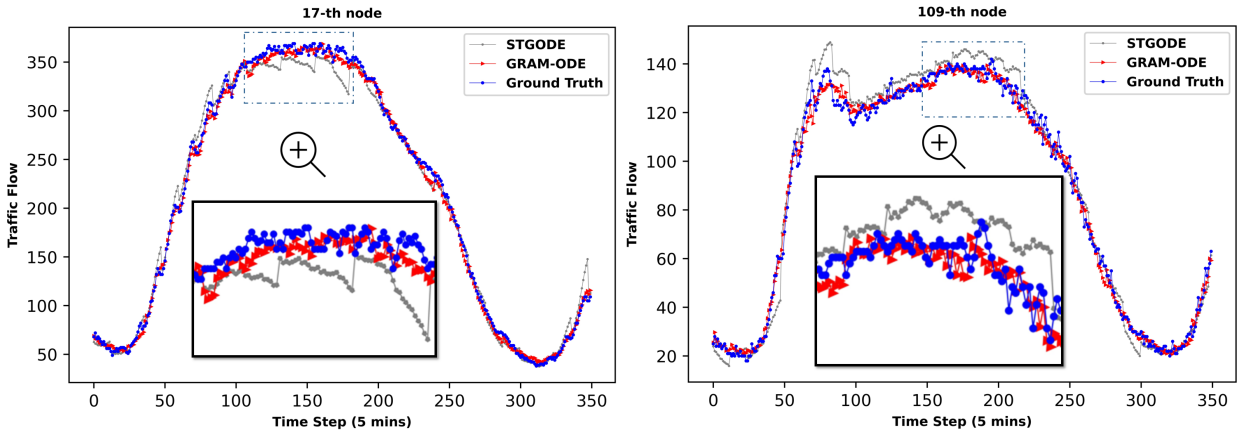


Figure 4: The comparison of traffic flow forecasting between our proposed GRAM-ODE and STGODE visualized for node 17 (left column) and node 109 (right column) of the PEMS08 dataset.

### 5.7 Ablation Study

To investigate the effect of different components of GRAM-ODE, we conduct ablation experiments on PEMS04 and PEMS08 with the several different variants of our model.

(1) **Base:** In this model, data only passes through a TCN, an ODE block and another TCN module. The ODE block only contains the global ODE-GNN with a fully connected layer after that. The output of TCN will be then imported to a simple fully connected layer instead of an attention module.

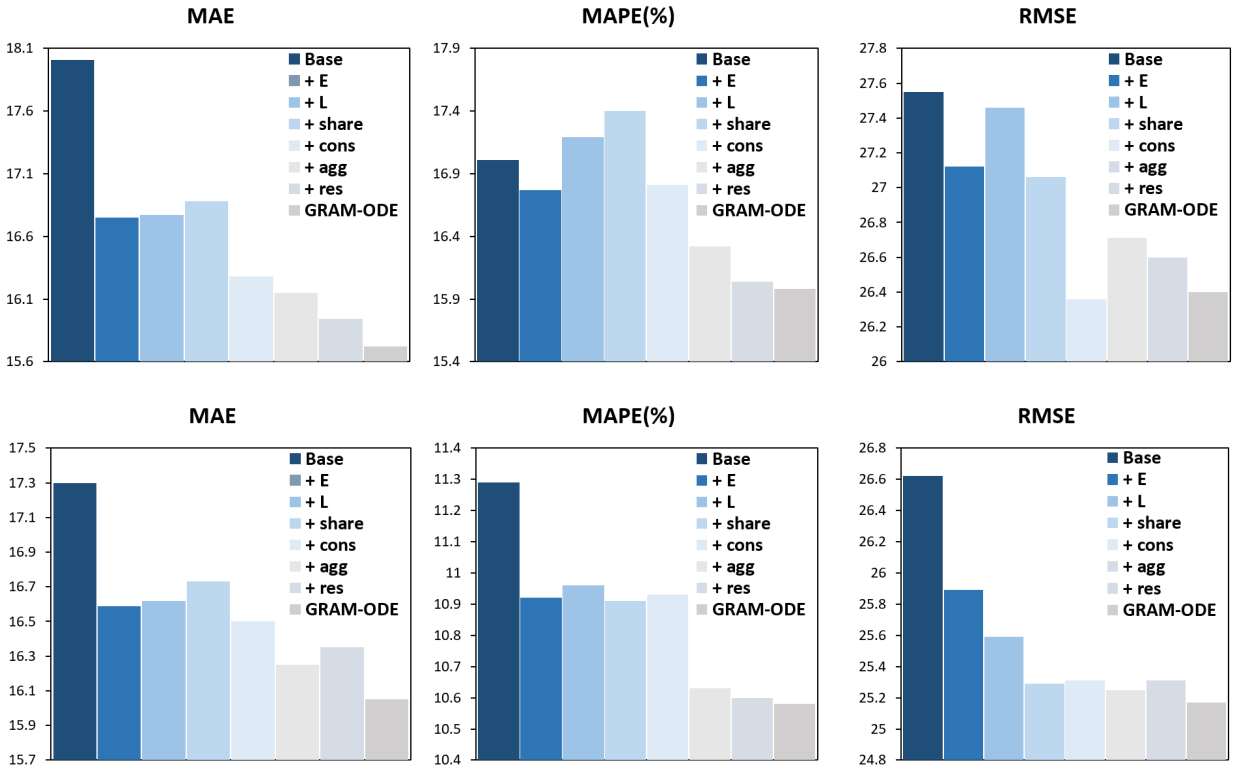


Figure 5: Ablation study experiments with different configurations of GRAM-ODE on PEMS03 (top row) and PEMS08 (bottom row) datasets.

(2) **+E:** Beyond (1), this model adds the dynamic edge correlations with an edge-based ODE-GNN block and aggregates its outputs with the global ODE-GNN block through a simple weighted sum operation.

(3) **+L:** Beyond (2), this model adds the local ODE-GNN block with different temporal kernels to better consider the local temporal patterns. The outputs of local ODE modules will be aggregated with other ODE modules through a weighted sum operation.

(4) **+share:** Compared to (3), this model uses shared temporal weights between the edge and global ODE-GNN modules and uses shared spatial weights among all three ODE-GNN modules, global, local and edge module.

(5) **+cons:** Beyond (4), this model adds an adaptive message filtering constraint to restrict the divergence of embeddings from local and global ODE modules.

(6) **+agg:** Beyond (5), this model replaces the weighted sum aggregation with a newly designed aggregation module explained in Eq. (19).

(7) **+res:** Beyond (6), this model adds the intra-block residual connections between outputs and inputs of the multi ODE-GNN blocks (which is explained in Eq. (20)).

(8) GRAM-ODE: Beyond (7), this model replaces the last linear layer with the attention module to better combine the features learned from different traffic graphs and parallel channels of GRAM-ODE layers (given by Eqs. (22) and (23)).

Fig. 5 shows the results of ablation experiments with MAE, MAPE, and RMSE metrics. It can be observed that the edge and local ODE-GNN modules are both enhancing the feature representation in the traffic

forecasting task. The model variant '+E' improves the performance of the base model across all metrics and both datasets. This shows that the simple node-based global ODE-GNN is insufficient in learning informative features and adding the edge-based global ODE-GNN module can considerably help the performance. Without any further techniques, the model gets worse by only using '+L' beyond '+E'. However, after adding the shared weight and divergence constraint techniques, the model usually gets better across all metrics. The shared weights are applied in the spatial and temporal operations of global node-based and edge-based ODE-GNN blocks to take into account the dynamic correlations of edges as well as nodes in the model formulation. The constraint is added to prevent local and global ODE-GNN embeddings deviating from one another. In this figure, we can also observe the impact of aggregation layer, residual-based update layer as well as the designed attention module. It appears that, among these three elements, adding the attention module will always result in better performance, which is consistent with our hypothesis that the attention mechanism makes the model more intelligent in considering all the high-dimensional correlations during feature extraction.

### 5.8 Robustness Study

To evaluate the robustness of our model, we add noise to the historical input of training data, which can potentially mimic uncertainties and biases that can arise during the data collection process. The added noise follows zero-mean i.i.d Gaussian distribution with fixed variance, e.g., $\mathcal{N}(0, \gamma^2)$, where $\gamma^2 \in \{2, 4\}$. We conduct robustness analysis across different values of $n \in \{0.1, 0.2, 0.3, 0.4\}$ representing the ratio of training data impacted by noise. In other words, $n = 0$ captures the performance without any noise. Fig. 6 represents the results of robustness comparisons between GRAM-ODE and STGODE on PEMS04 dataset across all the three metrics (MAE, MAPE, and RMSE). It can be observed that GRAM-ODE performance is more robust than STGODE with different levels of added noise $n = 0.1; 0.2; 0.3; 0.4$ which is probably due to the powerful spatio-temporal feature extraction gained by multiple ODE-GNN modules. We can also notice that, when the noise levels are high (with $\gamma^2 = 4$ and $n = 0.4$), GRAM-ODE can still beat many baseline models listed in Table 3, demonstrating the significant benefit of incorporating various ODE modules in our framework which can improve robustness.

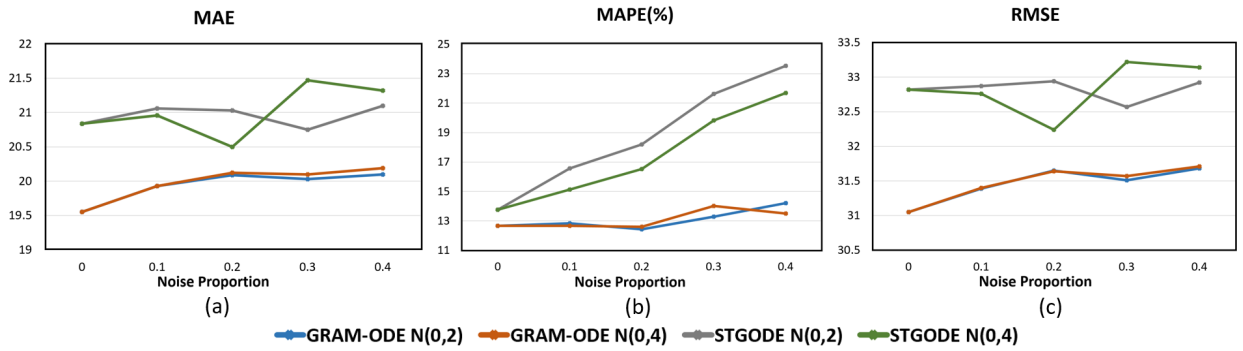


Figure 6: Robustness comparison of GRAM-ODE and STGODE on PEMS04 dataset.

## 6 Conclusion

In this paper, we propose Spatio-Temporal Graph Multi-ODE Neural Networks (GRAM-ODE) for the traffic forecasting task. In this model, multiple coupled ODE-GNN blocks are used to capture complex spatio-temporal dependencies from different views and learn better representations. We also add some techniques to further improve the communication between different ODE-GNN blocks including sharing weights, advanced aggregation, and divergence constraint. Extensive experiments on four real-world datasets show the superior performance of our proposed model compared to other state-of-the-art models as well as the effectiveness of each component in our model. Since our proposed framework is designed in a generic way, it can be applied to several other spatio-temporal forecasting applications.

## References

Taghreed Alghamdi, Khalid Elgazzar, Magdi Bayoumi, Taysseer Sharaf, and Sumit Shah. Forecasting traffic congestion using arima modeling. In *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*, pp. 1227–1232, 2019. doi: 10.1109/IWCMC.2019.8766698.

Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pp. 359–370. Seattle, WA, USA:, 1994.

George EP Box and David A Pierce. Distribution of residual autocorrelations in autoregressive-integrated moving average time series models. *Journal of the American statistical Association*, 65(332):1509–1526, 1970.

Ming Chen, Zhewei Wei, Zengfeng Huang, Bolin Ding, and Yaliang Li. Simple and deep graph convolutional networks. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1725–1735. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/chen20v.html.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. Spatial-temporal graph ode networks for traffic flow forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 364–373, 2021a.

Zheng Fang, Qingqing Long, Guojie Song, and Kunqing Xie. Spatial-temporal graph ode networks for traffic flow forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery Data Mining*, KDD '21, pp. 364–373, New York, NY, USA, 2021b. Association for Computing Machinery. ISBN 9781450383325. doi: 10.1145/3447548.3467430. URL https://doi.org/10.1145/3447548.3467430.

Shengnan Guo, Youfang Lin, Ning Feng, Chao Song, and Huaiyu Wan. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):922–929, Jul. 2019. doi: 10.1609/aaai.v33i01.3301922. URL https://ojs.aaai.org/index.php/AAAI/article/view/3881.

Mansura Habiba and Barak A. Pearlmutter. Neural ordinary differential equation based recurrent neural network model. 2020. URL https://arxiv.org/abs/2005.09807.

Zhixiang He, Chi-Yin Chow, and Jia-Dong Zhang. Stcnn: A spatio-temporal convolutional neural network for long-term traffic prediction. In *2019 20th IEEE International Conference on Mobile Data Management (MDM)*, pp. 226–233. IEEE, 2019.

Young-Seon Jeong, Young-Ji Byon, Manoel Mendonca Castro-Neto, and Said M. Easa. Supervised weighting-online learning algorithm for short-term traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 14(4):1700–1707, 2013. doi: 10.1109/TITS.2013.2267735.

Patrick Kidger. On neural differential equations. *CoRR*, abs/2202.02435, 2022. URL https://arxiv.org/abs/2202.02435.

Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 6696–6707. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/4a5876b450b45371f6cfe5047ac8cd45-Paper.pdf.

Patrick Kidger, James Foster, Xuechen Li, and Terry Lyons. Efficient and accurate gradients for neural SDEs. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021a. URL https://openreview.net/forum?id=b2bkE0Qq8Ya.

Patrick Kidger, James Foster, Xuechen Li, Harald Oberhauser, and Terry Lyons. Neural {sde}s made easy: {SDE}s are infinite-dimensional {gan}s, 2021b. URL https://openreview.net/forum?id=padYzanQNbg.

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

Mathias Lechner and Ramin Hasani. Mixed-memory RNNs for learning long-term dependencies in irregularly sampled time series, 2022.

Mengzhang Li and Zhanxing Zhu. Spatial-temporal fusion graph neural networks for traffic flow forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4189–4196, May 2021. doi: 10.1609/aaai.v35i5.16542. URL https://ojs.aaai.org/index.php/AAAI/article/view/16542.

Qimai Li, Zhichao Han, and Xiao-Ming Wu. Deeper insights into graph convolutional networks for semi-supervised learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018a. ISBN 978-1-57735-800-8.

Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In *International Conference on Learning Representations*, 2018b. URL https://openreview.net/forum?id=SJiHXGWAZ.

Jun Liu, Amir Shahroudy, Dong Xu, and Gang Wang. Spatio-temporal lstm with trust gates for 3d human action recognition. In *European conference on computer vision*, pp. 816–833. Springer, 2016.

James Morrill, Patrick Kidger, Lingyi Yang, and Terry Lyons. Neural controlled differential equations for online prediction tasks, 2021. URL https://arxiv.org/abs/2106.11028.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/bdbca288fee7f92f2bfa9f7012727740-Paper.pdf.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

Xingjian SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/07563a3fe3bbe7e3ba84431ad9d055af-Paper.pdf.

Chao Song, Youfang Lin, Shengnan Guo, and Huaiyu Wan. Spatial-temporal synchronous graph convolutional networks: A new framework for spatial-temporal network data forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 914–921, 2020.

Mascha Van Der Voort, Mark Dougherty, and Susan Watson. Combining kohonen maps with arima time series models to forecast traffic flow. *Transportation Research Part C: Emerging Technologies*, 4(5):307–318, 1996.

Vivek Veeriah, Naifan Zhuang, and Guo-Jun Qi. Differential recurrent neural networks for action recognition. In *Proceedings of the IEEE international conference on computer vision*, pp. 4041–4049, 2015.

Billy M Williams and Lester A Hoel. Modeling and forecasting vehicular traffic flow as a seasonal arima process: Theoretical basis and empirical results. *Journal of transportation engineering*, 129(6):664–672, 2003.

Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. Graph wavenet for deep spatial-temporal graph modeling. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, IJCAI'19, pp. 1907–1913. AAAI Press, 2021. ISBN 9780999241141.

SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pp. 802–810, 2015.

Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pp. 3634–3640. International Joint Conferences on Artificial Intelligence Organization, 7 2018. doi: 10.24963/ijcai.2018/505.

Junbo Zhang, Yu Zheng, and Dekang Qi. Deep spatio-temporal residual networks for citywide crowd flows prediction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1), Feb. 2017. doi: 10.1609/aaai.v31i1.10735. URL `https://ojs.aaai.org/index.php/AAAI/article/view/10735`.

Lun Zhang, Qiuchen Liu, Wenchen Yang, Nai Wei, and Decun Dong. An improved k-nearest neighbor model for short-term traffic flow prediction. *Procedia - Social and Behavioral Sciences*, 96:653–662, 2013. ISSN 1877-0428. doi: https://doi.org/10.1016/j.sbspro.2013.08.076. URL `https://www.sciencedirect.com/science/article/pii/S1877042813022027`. Intelligent and Integrated Sustainable Multimodal Transportation Systems Proceedings from the 13th COTA International Conference of Transportation Professionals (CICTP2013).

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1: 57–81, 2020. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2021.01.001. URL `https://www.sciencedirect.com/science/article/pii/S2666651021000012`.