Decoupled Dynamic Spatial-Temporal Graph Neural Network for Traffic Forecasting

Zezhi Shao[1,2], Zhao Zhang[1], Wei Wei[3,*], Fei Wang[1,*], Yongjun Xu[1], Xin Cao[4], Christian S. Jensen[5]
[1]Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2]University of Chinese Academy of Sciences, Beijing, China
[3]School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China
[4]School of Computer Science and Engineering, The University of New South Wales, Australia
[5]Department of Computer Science, Aalborg University, Denmark
{shaozezhi19b, zhaozhang2021, wangfei, xyj}@ict.ac.cn, weiw@hust.edu.cn, xin.cao@unsw.edu.au, csj@cs.aau.dk

ABSTRACT

We all depend on mobility, and vehicular transportation affects the daily lives of most of us. Thus, the ability to forecast the state of traffic in a road network is an important functionality and a challenging task. Traffic data is often obtained from sensors deployed in a road network. Recent proposals on spatial-temporal graph neural networks have achieved great progress at modeling complex spatial-temporal correlations in traffic data, by modeling traffic data as a diffusion process. However, intuitively, traffic data encompasses two different kinds of hidden time series signals, namely the diffusion signals and inherent signals. Unfortunately, nearly all previous works coarsely consider traffic signals entirely as the outcome of the diffusion, while neglecting the inherent signals, which impacts model performance negatively. To improve modeling performance, we propose a novel **D**ecoupled **S**patial-**T**emporal **F**ramework (DSTF) that separates the diffusion and inherent traffic information in a data-driven manner, which encompasses a unique estimation gate and a residual decomposition mechanism. The separated signals can be handled subsequently by the diffusion and inherent modules separately. Further, we propose an instantiation of DSTF, **D**ecoupled **D**ynamic **S**patial-**T**emporal **G**raph **N**eural **N**etwork (D^{2}STGNN), that captures spatial-temporal correlations and also features a dynamic graph learning module that targets the learning of the dynamic characteristics of traffic networks. Extensive experiments with four real-world traffic datasets demonstrate that the framework is capable of advancing the state-of-the-art.

PVLDB Reference Format:
Zezhi Shao, Zhao Zhang, Wei Wei, Fei Wang, Yongjun Xu, Xin Cao, and Christian S. Jensen. Decoupled Dynamic Spatial-Temporal Graph Neural Network for Traffic Forecasting. PVLDB, 15(11): 2733 - 2746, 2022.
doi:10.14778/3551793.3551827

PVLDB Artifact Availability:
The source code of this research paper has been made publicly available at https://github.com/zezhishao/D2STGNN.

1 INTRODUCTION

Traffic forecasting is a crucial service in Intelligent Transportation Systems (ITS) to predict future traffic conditions (*e.g.,* traffic flow[1]) based on historical traffic conditions [51] observed by sensors [9, 19]. This functionality fuels a wide range of services related to traffic management [4], urban computing [52], public safety [48], and beyond [14, 28, 34, 47].

Previous traffic forecasting studies usually fall into two categories, *i.e.,* knowledge-driven [3] and data-driven [7, 18, 21, 41]. The former commonly adopt queuing theory for user behavior simulation in traffic [3], while neglecting the natural complexity of real-world traffic flow. Regarding the latter, many early studies formulate the problem as a simple time series (*e.g.,* single variant time series) prediction task [20] and address it via various conventional statistic-based methods, such as auto-regressive integrated moving average (ARIMA [38]) and Kalman filtering [18]. These methods do not handle the high non-linearity of each time series well, since they typically rely heavily on stationarity-related assumptions. More importantly, they disregard the complex correlations among time series, which severely limits the effectiveness of traffic forecasting.

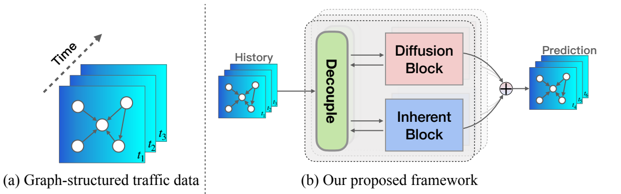
Recently, deep learning-based approaches [21, 24, 43] have been proposed to capture the complex spatial-temporal correlations in traffic flow. A promising and effective way is to construct an adjacency matrix to model the complex spatial topology of a road network and formulates the traffic data as a spatial-temporal graph. An example is shown in Figure 1(a), where each node represents a sensor, and the signals on each node vary over time. Sequentially, STGNN-based methods are proposed for traffic forecasting that models the dynamics of the traffic flow as a diffusion process [21, 41], and combines diffusion graph convolution [21] and sequential models [7, 46] to jointly model complex spatial-temporal correlations. The former [21] models the diffusion of vehicles among sensors in a road network, *i.e.,* the spatial dependency. The latter [7, 46] models the temporal dynamics, *i.e.,* the temporal dependency.

Although encouraging results have been achieved, these methods still fail to fully exploit the complex spatial-temporal correlations. First of all, each signal (*i.e.,* time series) naturally contains two different types of signals, *i.e.,* diffusion and non-diffusion signals (which is also called inherent signal for simplicity). The diffusion signal captures the vehicles diffused from other sensors, while the

[*] Corresponding author.

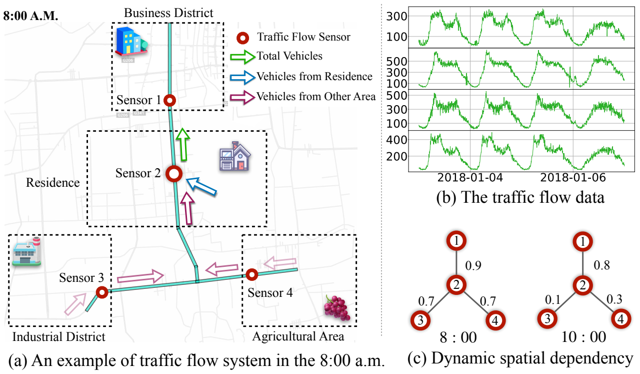
Proceedings of the VLDB Endowment, Vol. 15, No. 11 ISSN 2150-8097.
doi:10.14778/3551793.3551827

[1] An example of a traffic flow system is shown in Figure 2(a), where traffic sensors are deployed at important locations in the road network and record the total number of vehicles passing during a unit time interval. Over time, we can get four time series corresponding to sensors 1 to 4, as shown in Figure 2(b).



(a) Graph-structured traffic data (b) Our proposed framework

Figure 1: Graph structured traffic data and our proposed framework.



(a) An example of traffic flow system in the 8:00 a.m. (b) The traffic flow data (c) Dynamic spatial dependency

Figure 2: An example of the traffic flow system.

non-diffusion signal captures the vehicles that are independent of other sensors. However, almost all previous studies consider traffic data as a diffusion signal while disregarding the non-diffusion signal. That is, they model the complex spatial-temporal correlations coarsely. However, a reasonable solution is to exploit the complex spatial-temporal correlations more subtly, *i.e.,* explicitly modeling the diffusion and inherent signal simultaneously. Second, the predefined adjacency matrix in STGNN-based methods is static, and thus such construction methods may severely restrict the representative ability to complex road networks, making it difficult for these methods to model the dynamics of traffic flow. We illustrate them with examples in Figure 2. Without loss of generality, Figure 2 presents a typical traffic flow system. Important locations in the road network are equipped with traffic sensors that record traffic flow data, *i.e.,* the number of vehicles during a unit time interval. From Figure 2, we make two observations. **(I)** The recorded values of each sensor are affected by two factors, *i.e.,* the diffusion signal and the non-diffusion signal. As shown in Figure 2(a), vehicles passing through sensor 2 (green arrow) at 8 a.m. come from two parts. The first part is vehicles that depart directly from somewhere in the area near the sensor (blue arrow), *e.g.,* vehicles that drive directly from residence to the business district to work. The other part is vehicles diffused from adjacent areas (wine-red arrow), *e.g.,* vehicles that drive from the industrial district (sensor 3) and the agricultural area (sensor 4) to provide daily supplies. The former is independent of other sensors, while the latter is an artifact of the diffusion process. We call them *hidden inherent time series* and *hidden diffusion time series*, respectively, and each time series in Figure 2(b) is a superposition of them. **(II)** The traffic flow within the same road network may change over time, *i.e.,* spatial dependency is dynamic. An example is shown in Figure 2(c), where the traffic at sensors 3 and 4 can significantly affect sensor 2 at 8 a.m., while there is only a small influence at 10 a.m.

Therefore, addressing the above issues to effectively leverage all complex spatial-temporal correlations in traffic data is essential for improving the performance of traffic forecasting. To achieve this, we first propose a **D**ecoupled **S**patial-**T**emporal **F**ramework (DSTF) that is illustrated in the diagram in Figure 1(b). DSTF separates the diffusion and inherent traffic information using a the decouple block in a data-driven manner. Furthermore, we design a dynamic graph learning module based on a self-attention mechanism to address the second issue. The above designs are key elements of an instantiation of DSTF, called the **D**ecoupled **D**ynamic **S**patial-**T**emporal **G**raph **N**eural **N**etwork (D^{2}STGNN). Specifically, we first design the decouple block shown in Figure 3, which contains a residual decomposition mechanism and an estimation gate to decompose traffic data. The former removes the parts of signals that the diffusion and inherent models can approximate well. Thus, the parts of signals that are not learned well is retained. The latter estimates roughly the proportion of the two kinds of signals to relieve the burden of the first model in each layer, which takes the original signal as input and needs to learn specific parts in it. Second, the dynamic graph learning module comprehensively exploits available information to adjust the road network-based spatial dependency by learning latent correlations between time series based on the self-attention mechanism. In addition, specialized diffusion and inherent models, for the two hidden time series are designed according to their particular characteristics. A spatial-temporal localized convolution is designed to model the *hidden diffusion time series.* A recurrent neural network and self-attention mechanism are used jointly to model the *hidden inherent time series.*

In summary, the main contributions are the following:

- We propose a novel Decoupled Spatial-Temporal Framework (DSTF) for traffic forecasting, which decouples the hidden time series generated by the diffusion process and the hidden time series that is independent of other sensors. This enables more precise modeling of the different parts of traffic data to improve prediction accuracy.
- Based on the DSTF, a dynamic graph learning module is proposed that takes into account the dynamic nature of spatial dependency. Besides, we design a diffusion model and a inherent model to handle the two hidden time series. The above design forms our instantiation of DSTF, D^{2}STGNN.
- We conduct extensive experiments on four real-world, large-scale datasets to gain insight into the effectiveness of the framework DSTF and the instantiation D^{2}STGNN. Experimental results show that our proposal is able to consistently and significantly outperforms all baselines.

The paper is organized as follows. Section 2 covers related work, and Section 3 presents preliminaries and the problem definition. In Section 4, we present the decoupled spatial-temporal framework in detail. Section 5 details the chosen instantiation of the framework, D^{2}STGNN. We present extensive performance experiments and prediction visualizations in Section 6. We also report on extensive ablation studies of different architectures, important components, and training strategies. Section 7 concludes the paper.

2 RELATED WORK

With the availability of large-scale traffic data and the rise of artificial intelligence [42], Spatial-Temporal Graph Neural Networks (STGNNs) are proposed to model the complex spatial-temporal correlations in traffic data [21, 26, 29, 45, 49]. Generally speaking, STGNNs model the traffic system as a diffusion process [21] and combine the diffusion convolutions [21, 41] and sequential models [7, 46] to jointly model the spatial-temporal correlation. The diffusion convolutions are variants of Graph Convolution Networks (GCN [2, 8, 16]), which are well suited to deal with the non-euclidean relationships between multiple time series in traffic data. Sequential models, such as GRU [7], LSTM [32], and TCN [46], are used to model temporal dependency. For example, DCRNN [21] integrates diffusion convolution and the sequence to sequence architecture [33] to model the diffusion process. Graph WaveNet [41] combines diffusion convolution with dilated casual convolution [46] to capture spatial-temporal correlation efficiently and effectively.

Recent works focus on designing more powerful diffusion convolution models and sequential models. For example, many variants of GCNs, such as GAT [36], MixHop [1], and SGC [39], are adapted to STGNNs for better performance [6, 27, 31, 40, 44, 50]. The attention mechanism [35] and its variants, which theoretically have infinite receptive field size, are widely used to capture long-term temporal dependencies [12] in the sequential model [37, 53]. Moreover, a few very recent works propose to model the dynamic spatial dependency [13, 51]. The idea is to learn the latent correlations between nodes based on dynamic node feature, which is usually represented by the combination of real-time traffic features and other external features. For example, GMAN [51] designs a spatial attention mechanism by considering traffic features and node embeddings from the graph structure to learn the attention score.

Although STGNNs have made considerable progress, we find there is still significant room for improvement. Firstly, existing works solely consider the traffic data as a diffusion signal while neglecting the non-diffusion one, as discussed in Section 1. They model the complex spatial-temporal correlations in a coarse manner, which may impact model performance negatively. Secondly, although there are a few works on modeling dynamic spatial dependency, they do not consider all available information. Most of them explore the dynamic spatial dependency based on the feature of traffic conditions, ignoring either the constraints of the road network topology [13], or the time [51] or node [11] information.

3 PRELIMINARIES

We first define the notions of traffic network and traffic signal, and then define the forecasting problem addressed. Frequently used notations are summarized in Table 1.

Definition 1. ***Traffic Sensor.*** *A traffic sensor is a sensor deployed in a traffic system, such as a road network, and it records traffic information such as the flow of passing vehicles or vehicle speeds.*

Definition 2. ***Traffic Network.*** *A traffic network is a directed or undirected graph $G = (V, E)$, where V is the set of $|V| = N$ nodes and each node corresponds to a deployed sensor, and E is the set of $|E| = M$ edges. The reachability between nodes, expressed as an adjacent matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, could be obtained based on the pairwise road network distances between nodes.*

Table 1: Frequently used notation. Uppercase bold symbols denote 2D matrices. Calligraphic symbols denote 3D tensors.

Notations	Definitions
G	The traffic network $G = (V, E)$ with node set V and edge set E.
N	Number of sensors (nodes) of the traffic network, *i.e.,* $|V| = N$.
$\mathbf{A}$	The adjacency matrix of traffic network G.
T_h	The number of past traffic signals considered.
T_p	The number of future time steps to forecast.
C	Number of feature channels in a traffic signal.
d	Dimensionality of hidden states.
$\mathbf{E}$	Embedding of the sensors (nodes).
$\mathbf{T}$	Embedding of the time steps.
$\mathbf{W}$	Parameter matrix of the fully connected layer.
$\mathbf{X}_t$	Traffic signal at the t-th time step.
$\mathbf{H}_t$	Hidden state at the t-th time step.
$\mathcal{X}$	Traffic signals of the T_h most recent past time steps.
$\mathcal{Y}$	Traffic signals of the T_f nearest-future time steps.
$\mathcal{H}$	Hidden states over multiple time steps.
$\odot$	Element-wise product.
$\|$	Concatenation.
$Concat(\cdot)$	Broadcast concatenation.

Definition 3. ***Traffic Signal.*** *The traffic signal $\mathbf{X}_t \in \mathbb{R}^{N\times C}$ denotes the observation of all sensors on the traffic network G at time step t, where C is the number of features collected by sensors.*

Definition 4. ***Traffic Forecasting.*** *Given historical traffic signals $\mathcal{X} = [\mathbf{X}_{t-T_h+1}, \cdots, \mathbf{X}_{t-1}, \mathbf{X}_t] \in \mathbb{R}^{T_h\times N\times C}$ from the passed T_h time steps, traffic forecasting aims to predict the future traffic signals $\mathcal{Y} = [\mathbf{X}_{t+1}, \mathbf{X}_{t+2}, \cdots, \mathbf{X}_{t+T_f}]$ of the T_f nearest future time steps.*

4 THE DECOUPLED FRAMEWORK

The Decoupled Spatial-Temporal Framework (DSTF) that we propose is illustrated in Figure 3. Raw traffic signals are firstly transformed from the original space $\mathbb{R}^{T_h\times N\times C}$ to the latent space $\mathbb{R}^{T_h\times N\times d}$ by a linear layer. For simplicity, we use $\mathcal{X} \in \mathbb{R}^{T_h\times N\times d}$ in the following as default. DSTF contains multiple decoupled spatial-temporal layers. Given traffic signals $\mathcal{X} \in \mathbb{R}^{T_h\times N\times d}$, the decoupled spatial-temporal layer aims at decomposing them into two hidden signals: $\mathcal{X} = \mathcal{X}^{dif} + \mathcal{X}^{inh}$, where $\mathcal{X}^{dif}$ and $\mathcal{X}^{inh}$ denote the diffusion signals and the inherent signals, respectively. However, it is a challenging task to separate them since we do not have prior knowledge. To this end, we propose a residual decomposition mechanism and an estimation gate in the decouple block (the green block) to decompose the spatial-temporal signals in a data-driven manner.

4.1 Residual Decomposition Mechanism

We first design the residual decomposition mechanism, which decomposes the traffic signals by removing the part that has been

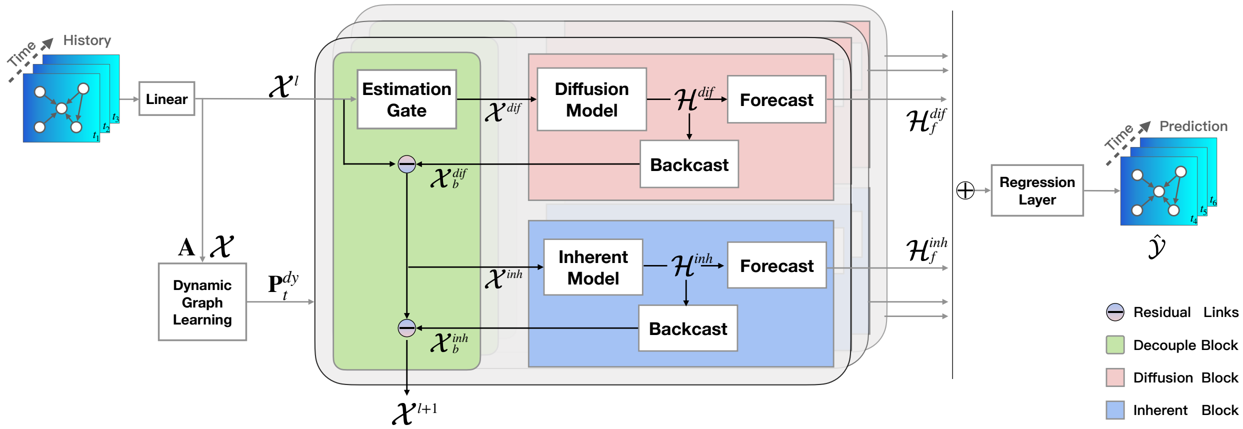


Figure 3: The overall architecture of the proposed D^{2}STGNN. The decouple block (green) decomposes each time series in traffic signals into two hidden time series, which are subsequently handled by the diffusion block (pink) and inherent block (blue). Moreover, the dynamic graph learning module generates dynamic spatial dependency for the diffusion model.

learned by the diffusion model or inherent model in an information reconstruction fashion. As shown in Figure 3, except for the decouple block (green), the decoupled spatial-temporal layer contains a diffusion block (pink) and an inherent block (blue), each with three components: a primary model that learns knowledge from the input data $\mathcal{X}^* \in \mathbb{R}^{T_h \times N \times d}$ and generates hidden states $\mathcal{H}^* \in \mathbb{R}^{T_h \times N \times d}$, a forecast branch that generates the module's forecast hidden state $\mathcal{H}_f^*$, a backcast branch that generates the best estimate of the module's input signal $\mathcal{X}_b^* \in \mathbb{R}^{T_h \times N \times d}$. The star indicates that this applies to both the diffusion and the inherent blocks.

The backcast branch is crucial for the decoupling, since it reconstructs the learned knowledge, *i.e.,* the portion of input signals that the current model can approximate well. Subsequently, residual links are designed to remove the signals that can be approximated well from the input signals and retain the signals that are not well decomposed. Therefore, after the first (upper) residual link, we get the input of the inherent block, *i.e.,* the inherent signals:

$$\mathcal{X}^{inh} = \mathcal{X}^l - \mathcal{X}_b^{dif} = \mathcal{X}^l - \sigma(\mathcal{H}^{dif}\mathbf{W}_b^{dif}) \tag{1}$$

where $\mathcal{X}^l$ is the input of (l)-th layer, and $\mathcal{X}^0 = \mathcal{X}$. Superscripts *dif* and *inh* indicate diffusion and inherent information, respectively. We use non-linear fully connected networks to implement the backcast branch. $\mathbf{W}_b^{dif}$ is the network parameters, and the σ is the ReLU [10] activation function. Similarly, we conduct the second (lower) residual link after the inherent block:

$$\mathcal{X}^{l+1} = \mathcal{X}^{inh} - \mathcal{X}_b^{inh} = \mathcal{X}^{inh} - \sigma(\mathcal{H}^{inh}\mathbf{W}_b^{inh}) \tag{2}$$

where $\mathcal{X}^{l+1}$ retains the residual signals that can not be decomposed in the l-th layer. Similar to other deep learning methods, we stack multiple decoupled spatial-temporal layers to enhance the model's capabilities, as shown in Figure 3. The spatial-temporal signal can be decoupled if we design proper models for diffusion and inherent signals according to their own particular characteristics, and each model can focus on its specific signals.

4.2 Estimation Gate

Although the residual decomposition can decouple traffic signals in a data-driven manner, the first model (the diffusion model in Figure 3) in each layer still faces a challenge that may fail the decoupling process: it takes original traffic data as input, but it needs to learn only the specific part of signals in it. To address this problem, the estimation gate is designed to reduce the burden of the first model by roughly estimating the proportion of the two hidden time series. At its core, the estimation gate learns a gate value automatically in $(0, 1)$ based on the current node and current time embeddings. Firstly, to take into account real-world periodicities, we utilize two time slot embedding matrices: $\mathbf{T}^D \in \mathbb{R}^{N_D \times d}$ and $\mathbf{T}^W \in \mathbb{R}^{N_W \times d}$, where N_D is the number of time slots of the day (determined by the sampling frequency of sensors) and $N_W = 7$ is the number of days in a week. The embeddings of time slots are thus shared among slots for the same time of the day and the same day of the week. Secondly, we use two matrices for node embeddings, the source node embedding: $\mathbf{E}^u \in \mathbb{R}^{N \times d}$ is used when a node passes messages to neighboring nodes, and the target node embedding $\mathbf{E}^d \in \mathbb{R}^{N \times d}$ is used when a node aggregates information from neighboring nodes. Kindly note that the node embeddings and time slot embeddings are randomly initialized with learnable parameters. Then, given the historical traffic data $\mathcal{X}$ and embeddings of time slots and nodes, the estimation gate can be written as:

$$\begin{gathered}\Lambda_{t,i} = Sigmoid(\sigma((\mathbf{T}_t^D \parallel \mathbf{T}_t^W \parallel \mathbf{E}_i^u \parallel \mathbf{E}_i^d)\mathbf{W}_1)\mathbf{W}_2) \\ \mathcal{X}^{dif} = \Lambda \odot \mathcal{X}^l\end{gathered} \tag{3}$$

where $\mathbf{\Lambda} \in \mathbb{R}^{T_h \times N \times 1}$, and $\mathbf{\Lambda}_{t,i} \in (0, 1)$ estimates the proportion of the diffusion signal in the time series in traffic data at time slot t of node i. The symbol $\odot$ denotes the element-wise product that broadcasts to all the channels of $\mathcal{X} \in \mathbb{R}^{T_h \times N \times d}$. $\mathbf{W}_1 \in \mathbb{R}^{4d \times d}$ and $\mathbf{W}_2 \in \mathbb{R}^{d \times 1}$ are learnable parameters, and σ is a non-linear activation function, such as ReLU [10].

In addition, although we use the example of Figure 3, where the diffusion block precedes the inherent block, they are in principle interchangeable since there is no significant difference in which signal is decomposed first. We conduct experiments in Section 6.5 to verify that there is no significant difference in performance. In this paper, we'll still keep the diffusion-first style. Besides, we omit the superscript l for each symbol of the decouple block except the input signal $\mathcal{X}^l$ and residual signal $\mathcal{X}^{l+1}$ for simplicity.

In summary, this section proposes a novel framework DSTF, where each time series in traffic data is decoupled to the diffusion signals and the inherent signals in a data-driven manner. Kindly note that other components, *i.e.,* dynamic graph learning, diffusion model, and inherent model, remain abstract and can be designed independently in the framework according to the characteristics of diffusion and inherent signals. In the next section, we give an instantiation of DSTF by carefully designing these components.

5 DECOUPLED DYNAMIC ST-GNN

By decomposing diffusion and inherent signals, the framework enables the subsequent models to focus each on what they do best. Here, we propose our Decoupled Dynamic Spatial-Temporal Graph Neural Network (D^{2}STGNN) as an instantiation of the proposed framework. We cover the details of the diffusion and inherent blocks as well as the dynamic graph learning shown in Figure 3.

5.1 Diffusion Model: Spatial-Temporal Localized Convolutional Layer

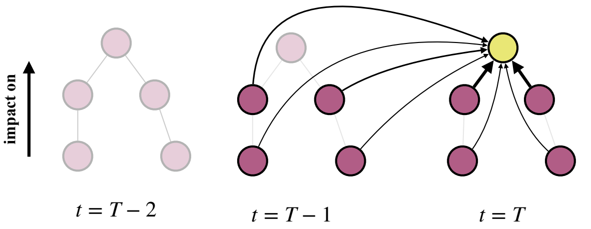


Figure 4: An example of spatial-temporal locality where $k_s = k_t = 2$. Only recent traffic signals from neighboring nodes can diffuse to a target node.

The diffusion model aims to model the diffusion process between nodes, where the future diffusion signals of a target node depend on the recent values of neighboring nodes, *i.e.,* the setting exhibits spatial-temporal locality. Specifically, we assume that only the traffic signals of k_s order neighboring nodes from the past k_t time steps can affect a target node. Considering the speed of vehicles and the sampling frequency of sensors, the typical values of k_s and k_t are usually 2 or 3. An example is shown in Figure 4, where $k_s = k_t = 2$. In order to capture such the diffusion process, we design a spatial-temporal localized convolutional layer.

Firstly, we define a spatial-temporal localized transition matrix:

$$(\mathbf{P}^{local})^k = \underbrace{(\mathbf{P}^k \odot (1 - \mathbf{I}_N)) \parallel \cdots \parallel (\mathbf{P}^k \odot (1 - \mathbf{I}_N))}_{k_t} \tag{4}$$

where $\mathbf{P}^k \in \mathbb{R}^{N \times N}$ is a k order transition matrix, and $k = 1, \cdots, k_s$. Given the road network adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, there are two directions of information diffusion: the forward transition $\mathbf{P}_f = \mathbf{A}/rowsum(\mathbf{A})$, the backward transition $\mathbf{P}_b = \mathbf{A}^T/rowsum(\mathbf{A}^T)$. Therefore, $(\mathbf{P}^{local})^k \in \mathbb{R}^{N \times k_t N}$, and $(\mathbf{P}^{local})^k[i, j]$ describes the influence of node j on node i in a localized spatial-temporal range. Note that $\mathbf{P}^{local}[i, i + k'N]$ $(k' = 0, 1, \cdots, k_t - 1)$ are masked to zeros since they describe the inherent patterns of the target nodes themselves, which will be learned by the inherent model. For simplicity, we abbreviate $(\mathbf{P}^{local})^k$ as $(\mathbf{P}^{lc})^k$. Secondly, corresponding to the Eq. 4, there is a localized feature matrix $\mathbf{X}_t^{lc} \in \mathbb{R}^{k_t N \times d}$ formed as:

$$\mathbf{X}_t^{lc} = [\underbrace{\sigma(\mathbf{X}_{t-k_t+1}^{dif}\mathbf{W}_{k_t-1})^T \parallel \cdots \parallel \sigma(\mathbf{X}_t^{dif}\mathbf{W}_0)^T}_{k_t}]^T \tag{5}$$

where $\mathbf{W}_k$ is the learnable parameter and σ is the ReLU activation function. The non-linear transformation used here aims to strengthen the expressive power of the model.

Therefore, based on the transition matrix $(\mathbf{P}^{lc})^k$ and feature matrix $\mathbf{X}_t^{lc}$ mentioned above, we define our spatial-temporal localized graph convolution operator with spatial kernels size k_s as:

$$\mathbf{H}_t^{dif} = \sum_{k=1}^{k_s} (\mathbf{P}^{lc})^k \mathbf{X}_t^{lc} \mathbf{W}_k \tag{6}$$

where $\mathbf{H}_t^{dif} \in \mathbb{R}^{N \times d}$ is the output of the localized graph convolution operator at time step t, which considers spatial information from the k_s order neighbors. Next, $\mathbf{W}_k$ is the graph convolution parameters of k-th order, and $\mathbf{H}_t^{dif}$ is the hidden state of subsequent time slots that can be used to predict the diffusion part.

In addition to the road network-based transition matrices $\mathbf{P}_b$ and $\mathbf{P}_f$, we also utilize a self-adaptive transition matrix [41]. Different from the transition matrices $\mathbf{P}_b$ and $\mathbf{P}_f$, which are handcrafted by prior human knowledge, the self-adaptive transition matrix is optimized by two randomly initialized node embedding dictionaries with learnable parameters $\mathbf{E}^u \in \mathbb{R}^{N \times d}$ and $\mathbf{E}^d \in \mathbb{R}^{N \times d}$:

$$\mathbf{P}_{apt} = Softmax(\sigma(\mathbf{E}^d(\mathbf{E}^u)^T)). \tag{7}$$

Note that the $\mathbf{P}_{apt} \in \mathbb{R}^{N \times N}$ is normalized by the *Softmax* function. Therefore, it describes the diffusion process that is similar to transition matrix $\mathbf{P}_b$ and $\mathbf{P}_f$. Indeed, the matrix $\mathbf{P}_{apt}$ can serve as supplement to of the hidden diffusion process that is missed in the road network-based transition matrices $\mathbf{P}_b$ and $\mathbf{P}_f$.

Given the three transition matrices $\mathbf{P}_f$, $\mathbf{P}_b$, and $\mathbf{P}_{apt}$, we can get their corresponding spatial-temporal localized transition matrix in Eq. 4, $(\mathbf{P}_f^{lc})^k$, $(\mathbf{P}_b^{lc})^k$, and $(\mathbf{P}_{apt}^{lc})^k$. We can now present our localized convolutional layer based on the operation in Eq. 6 as follows:

$$\mathbf{H}_t^{spa} = \sum_{k=1}^{k_s} \left[(\mathbf{P}_f^{lc})^k \mathbf{X}_t^{lc} \mathbf{W}_{k1} + (\mathbf{P}_b^{lc})^k \mathbf{X}_t^{lc} \mathbf{W}_{k2} + (\mathbf{P}_{apt}^{lc})^k \mathbf{X}_t^{lc} \mathbf{W}_{k3} \right]. \tag{8}$$

In summary, given temporal kernel size k_t and spatial kernel size k_s as well as input $\mathcal{X}^{dif} \in \mathbb{R}^{T_h \times N \times d}$, the localized convolutional layer generates a hidden state sequence $\mathcal{H}^{dif}$ by synchronously modeling the spatial-temporal correlations in each time step t:

$$\mathcal{H}^{dif} = \mathbf{\Theta}_{*G}(\mathcal{X}^{dif}) = [\cdots, \mathbf{H}_{T-2}^{dif}, \mathbf{H}_{T-1}^{dif}, \mathbf{H}_T^{dif}] \tag{9}$$

where $\mathbf{\Theta}$ denotes all the parameters mentioned in Eqs. 5, 7, and 8. G denotes the spatial-temporal localized convolution in Eq. 8. The output hidden state sequence $\mathcal{H}^{dif}$ is further used to generate two outputs, *i.e.,* the backcast and forecast output.

Forecast Branch: The last hidden state $\mathbf{H}_T^{dif}$ can be used to forecast the value of the next step. In order to forecast the hidden state in multi-step forecasting task, we follow an auto-regressive procedure to generate $\mathcal{H}_f^{dif} = [\mathbf{H}_{T+1}^{dif}, \mathbf{H}_{T+2}^{dif}, \dots, \mathbf{H}_{T+T_f}^{dif}]$, each of which is used by a non-linear regression neural network to predict the particular values that we are interested in.

Backcast Branch: As discussed in Section 4.1, we use non-linear fully connected networks to implement the backcast branch, and generate $\mathcal{X}_b^{dif} = \sigma(\mathcal{H}^{dif}\mathbf{W}_b^{dif})$, *i.e.,* the learned diffusion part, which subsequently is removed from the original signals by the residual link in Eq. 1 to achieve decomposition.

5.2 Inherent Model: Local and Global Dependency

The inherent model is designed to model the *hidden inherent time series* in the original signals of each node, *i.e.,* the $\mathcal{X}^{inh}$. Dependencies in time series are often divided into local and global dependencies, a.k.a. short- and long-term dependencies [17, 25, 37]. Previous studies have shown that Gated Recurrent Units [7] (GRUs) are good at capturing short-term dependencies, while a self-attention layer does better at handling long-term dependencies [37]. We utilize GRU [7] and a multi-head self-attention layer [35] jointly to capture temporal patterns comprehensively. A diagram of the inherent model is shown in Figure 5.

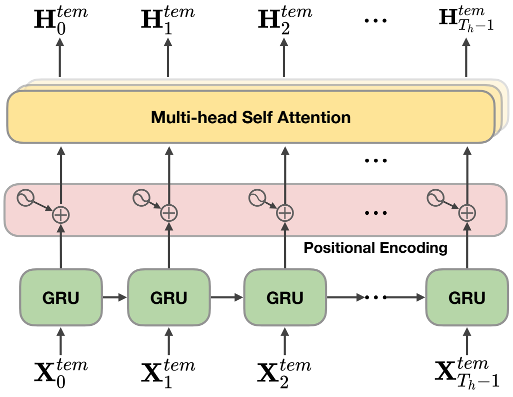


Figure 5: The inherent model. Short-term dependencies are captured by the GRU, while long-term dependencies are captured by the multi-head self-attention layer.

GRU can recurrently preserve the hidden state of history data and control the information that flows to the next time step. Given the input signal of inherent block $\mathbf{X}_t^{inh} \in \mathbb{R}^{N \times d}$ at time step t , for each node i, we use the following GRU operation:

$$\begin{aligned}
\mathbf{z}_t &= \sigma(\mathbf{W}_z\mathbf{X}_t^{inh}[i,:] + \mathbf{U}_z\mathbf{H}_{t-1}^{inh}[i,:] + \mathbf{b}_z) \\
\mathbf{r}_t &= \sigma(\mathbf{W}_r\mathbf{X}_t^{inh}[i,:] + \mathbf{U}_r\mathbf{H}_{t-1}^{inh}[i,:] + \mathbf{b}_r) \\
\hat{\mathbf{H}}_t^{inh}[i,:] &= tanh(\mathbf{W}_h\mathbf{X}_t^{inh}[i,:] + \mathbf{r}_t \odot (\mathbf{U}_h\mathbf{H}_{t-1}^{inh}[i,:] + \mathbf{b}_h)) \\
\widetilde{\mathbf{H}}_t^{inh}[i,:] &= (1 - \mathbf{z}_t) \odot \hat{\mathbf{H}}_{t-1}^{inh}[i,:] + \mathbf{z}_t \odot \hat{\mathbf{H}}_t^{inh}[i,:]
\end{aligned} \tag{10}$$

where $\widetilde{\mathbf{H}}_t^{inh}[i,:]$ is the updated hidden state of node i at time step t, $\odot$ denotes the element-wise product, and $\mathbf{W}_z$, $\mathbf{W}_r$, $\mathbf{W}_h$, $\mathbf{U}_z$, $\mathbf{U}_r$, and $\mathbf{U}_h$ are the learnable parameters of GRU.

The GRU can capture short-term sequential information well. However, capturing only local information is insufficient because traffic forecasting is also affected by longer-term dependencies [37]. Hence we introduce a multi-head self-attention layer to capture global dependencies on the top of the GRU. Given the output of the GRU, $\widetilde{\mathcal{H}}^{inh} \in \mathbb{R}^{T_h \times N \times d}$, the multi-head self-attention layer performs pair-wisely dot product attention on the time dimension for each node, *i.e.,* the product is calculated between any two signals of different time slots. Therefore, the receptive field is theoretically infinite, which is beneficial to capturing global dependencies. Specifically, given attention head s, the learnable project matrices $\mathbf{W}_s^Q, \mathbf{W}_s^K, \mathbf{W}_s^V \in \mathbb{R}^{d \times d}$, and the output matrix $\mathbf{W}^O$, the attention function of node i can be written as:

$$\begin{aligned}
\mathcal{H}^{inh}[:,i,:] &= Multihead(\mathbf{H}^{v_i}) \\
&= Concat(head_1, \cdots, head_S)\mathbf{W}^O \\
\text{where } head_s &= Attention_s(\mathbf{H}^{v_i}) \\
&= softmax(\frac{\mathbf{H}^{v_i}\mathbf{W}_s^Q(\mathbf{H}^{v_i}\mathbf{W}_s^K)^T}{\sqrt{d}}\mathbf{H}^{v_i}\mathbf{W}_s^V)
\end{aligned} \tag{11}$$

where $\mathbf{H}^{v_i} \in \mathbb{R}^{T \times d}$ is the feature of node v_i in all time slots. All the nodes are calculated individually in parallel with the help of the GPU. Hence, we can get the hidden state of inherent model $\mathcal{H}^{inh} \in \mathbb{R}^{T \times N \times d}$. Although the self-attention layer has an infinite receptive field, it ignores relative positions in the sequence. To take into account the position, we apply positional encoding [35] between GRU and multi-head self-attention layer as follows:

$$\begin{aligned}
\widetilde{\mathbf{H}}_t^{inh}[i,:] &= \widetilde{\mathbf{H}}_t^{inh}[i,:] + \mathbf{e}_t \\
\mathbf{e}_{t,i} &= \begin{cases} sin(t/10000^{2i/d}), & if\ i = 0, 2, 4... \\ cos(t/10000^{2i/d}), & otherwise \end{cases}
\end{aligned} \tag{12}$$

where $\mathbf{e}_t \in \mathbb{R}^d$ is the positional embedding of time step t. Note that the positional encoding is not trainable.

Forecast Branch: Here, we also adopt auto-regression to generate the future hidden state $\mathcal{H}_f^{inh} = [\mathbf{H}_{T+1}^{inh}, \mathbf{H}_{T+2}^{inh}, \dots, \mathbf{H}_{T+T_f}^{inh}]$. Specifically, we adopt a simple sliding auto-regression, rather than the commonly used encoder-decoder architecture [33, 35] because we do not have the ground truth of *hidden inherent time series*, which are crucial when having to train a decoder.

Backcast Branch: Same as the diffusion block, we use non-linear fully connected networks to implement the backcast branch and generate $\mathcal{X}_b^{inh} = \sigma(\mathcal{H}^{inh}\mathbf{W}_b^{inh})$, *i.e.,* the learned inherent part, which is subsequently used in Eq. 2.

5.3 Dynamic Graph Learning

In this subsection, we design a dynamic graph learning model to capture the dynamics of spatial dependency, as discussed in Section 1. The intensity of traffic diffusion between two connected nodes changes dynamically over time in the real world. An example is shown in Figure 2(c), where the influence between nodes is different between 8:00 am and 10:00 am. Therefore, it is crucial to model dynamic transition matrices $\mathbf{P}_b^{dy}$ and $\mathbf{P}_f^{dy}$ to enhance the static ones $\mathbf{P}_b$ and $\mathbf{P}_f$ by replacing them in Eqs. 8 and 4.

The core of modeling $\mathbf{P}_b^{dy}$ and $\mathbf{P}_f^{dy}$ is to ensure that the static, dynamic, and time information in traffic data is encoded comprehensively. For a given time step t, we take the historical observation as the dynamic feature. For example, given historical data $\mathcal{X} \in \mathbb{R}^{T_h \times N \times d}$, the dynamic information of channel c can be formulated as $\mathbf{X}_c = \mathcal{X}[:,:,c]^T \in \mathbb{R}^{N \times T}$, where $c = 1, ..., d$. In addition, we consider the time embeddings $\mathbf{T}_t^D \in \mathbb{R}^d$ and $\mathbf{T}_t^W \in \mathbb{R}^d$, which are the embeddings used in the estimation gate in Section 4.2. Employing also two static node embedding matrices $\mathbf{E}^u \in \mathbb{R}^{N \times d}$ and $\mathbf{E}^d \in \mathbb{R}^{N \times d}$, we first obtain two dynamic feature matrices:

$$\begin{aligned} \mathbf{DF}_t^u &= Concat[\mathrm{FC}(\overset{C}{\underset{c=1}{\|}} \mathbf{X}_c), \mathbf{T}_t^D, \mathbf{T}_t^W, \mathbf{E}^u] \\ \mathbf{DF}_t^d &= Concat[\mathrm{FC}(\overset{C}{\underset{c=1}{\|}} \mathbf{X}_c), \mathbf{T}_t^D, \mathbf{T}_t^W, \mathbf{E}^d]. \end{aligned} \tag{13}$$

Here, $\mathbf{DF}_t^u \in \mathbb{R}^{N \times 4d}$, and $\mathbf{DF}_t^d \in \mathbb{R}^{N \times 4d}$. And $\mathrm{FC}(\cdot)$ is a non-linear two-layer fully connected network that extracts features and transforms the dimensionality from $N \times dT$ to $N \times d$. Further, $Concat(\cdot)$ denotes broadcast concatenation. Then we use the attention mechanism to calculate the pair-wise mask to get dynamic graphs:

$$\begin{aligned} \mathbf{P}_{f,t}^{dy} &= \mathbf{P}_f \odot Softmax(\frac{(\mathbf{DF}_t^u \mathbf{W}^Q)(\mathbf{DF}_t^u \mathbf{W}^K)^T}{\sqrt{d}}) \\ \mathbf{P}_{b,t}^{dy} &= \mathbf{P}_b \odot Softmax(\frac{(\mathbf{DF}_t^d \mathbf{W}^Q)(\mathbf{DF}_t^d \mathbf{W}^K)^T}{\sqrt{d}}). \end{aligned} \tag{14}$$

$\mathbf{W}_Q$ and $\mathbf{W}_K$ are the parameters of self-attention mechanism [35]. Matrices $\mathbf{P}_{f,t}^{dy}$ and $\mathbf{P}_{f,t}^{dy} \in \mathbb{R}^{N \times N}$ can replace the transition matrices in Eqs. 8 and 4 to enhance the model, thus completing the proposed D^{2}STGNN model. In practice, the calculation of the adjacency matrix is expensive, so to reduce the computational cost, we assume that given a limited time range T_h, $\mathbf{P}^{dy}$ is static, *i.e.,* $\mathbf{P}_{t-T_h:t}^{dy} = \mathbf{P}_t^{dy}$.

5.4 Output and Training Strategy

Assuming we stack L decoupled spatial-temporal layers, we include the output hidden states in the forecast branches $\mathcal{H}_f^{dif,l} = [\mathbf{H}_{T+1}^{dif,l}, \cdots]$ and $\mathcal{H}_f^{inh,l} = [\mathbf{H}_{T+1}^{inh,l}, \cdots]$ of the diffusion and the inherent blocks of the l-th layer to generate our final forecasting:

$$\begin{aligned} \mathcal{H} &= \mathcal{H}_f^{dif} + \mathcal{H}_f^{inh} = \sum_{l=0}^{L-1} \mathcal{H}_f^{dif,l} + \sum_{l=0}^{L-1} \mathcal{H}_f^{inh,l} \\ &= [\sum_{l=0}^{L-1} (\mathbf{H}_{T+1}^{dif,l} + \mathbf{H}_{T+1}^{inh,l}), \sum_{l=0}^{L-1} (\mathbf{H}_{T+2}^{dif,l} + \mathbf{H}_{T+2}^{inh,l}), \cdots]. \end{aligned} \tag{15}$$

Then we adopt a two-layer fully connected network as our regression layer and apply it to $\mathcal{H}$ to generate the final predictions. The outputs of the regression layer at each time step are concatenated to form the final output: $\hat{\mathcal{Y}} \in \mathbb{R}^{T_f \times N \times C_{out}}$. Given the ground truth $\mathcal{Y} \in \mathbb{R}^{T_f \times N \times C_{out}}$, we optimize our model using MAE loss:

$$\mathcal{L}(\hat{\mathcal{Y}}, \mathcal{Y}; \Theta) = \frac{1}{T_f N C_{out}} \sum_{i=1}^{T_f} \sum_{j=1}^{N} \sum_{k=1}^{C_{out}} |\hat{\mathcal{Y}}_{ijk} - \mathcal{Y}_{ijk}| \tag{16}$$

where N is the number of nodes, T_f is the number of forecasting steps, and C_{out} is the dimensionality of the output. Following existing studies [20, 40], we employ curriculum learning, a general and effective training strategy, to train the proposed model. We optimize the model parameters by minimizing $\mathcal{L}$ via gradient descent. The overall learning algorithm is outlined in Algorithm 1.

Algorithm 1: The overall learning algorithm of D^{2}STGNN

```
Input: The traffic signals over the past $T_h$ time steps $\mathcal{X}$, the
       adjacency matrix $\mathbf{A}$, time embeddings $\mathbf{T}^D$ and $\mathbf{T}^W$,
       node embeddings $\mathbf{E}^d$ and $\mathbf{E}^u$, the number of layers $L$.
Output: The prediction of traffic signals $\mathcal{Y}$ in $T_f$ future
        time steps.
Calculate self-adaptive adjacent matrix $\mathbf{A}_{apt}$ by Eq. 7.
Calculate dynamic transition matrix $\mathbf{P}_f^{dy}$ and $\mathbf{P}_b^{dy}$ by Eq. 14.
output ← [ ]
$\mathcal{X}^0 \leftarrow \mathcal{X}$
for $l$ in range(L) do
    Calculate $\mathcal{X}^{dif}$ according to Eq. 3 with time and node
      embeddings.                                   ▹ Estimation gate
    Calculate $\mathcal{H}^{dif}, \mathcal{X}_b^{dif}$ according to Eq. 9 and the backcast
      branch.                                       ▹ Diffusion block
    Calculate $\mathcal{X}^{inh}$ according to Eq. 1.        ▹ Decomposition
    Calculate $\mathcal{H}^{inh}, \mathcal{X}_b^{inh}$ according to Eq. 11 and the
      backcast branch.                              ▹ Inherent block
    Calculate $\mathcal{X}^{l+1}$ according to Eq. 2.       ▹ Decomposition
    Append the output of forecast branch of diffusion and
      inherent block to the output list.
end
$\mathcal{H} \leftarrow$ sum(output)
$\hat{\mathcal{Y}} \leftarrow \mathrm{MLP}(\mathcal{H})$
Backpropagation and update parameters according to Eq. 16.
```

6 EXPERIMENTS

In this section, we present experiments on four large real-world datasets to demonstrate the effectiveness of D^{2}STGNN for traffic forecasting. We first introduce the experimental settings, including datasets, baselines, and parameter settings. Then, we conduct experiments to compare the performance of the D^{2}STGNN with other baselines. Furthermore, we design more experiments to verify the superiority of our decoupling framework. Finally, we design comprehensive ablation studies to evaluate the impact of the essential architectures, components, and training strategies.

6.1 Experimental Setup

Datasets. We conducted experiments on four commonly used real-world large-scale datasets, which have tens of thousands of time steps and hundreds of sensors. The statistical information is summarized in Table 2. Two of them are traffic speed datasets, while the others are traffic flow dataset. Traffic speed data records the average vehicles speed (miles per hour). Due to the speed limit in these areas, the speed data is a float value usually less than 70. The flow data should be an integer, up to hundreds, because it records

Table 2: Statistics of datasets.

Type	Dataset	# Node	# Edge	# Time Step
Speed	METR-LA	207	1722	34272
	PEMS-BAY	325	2694	52116
Flow	PEMS04	307	680	16992
	PEMS08	170	548	17856

the number of passing vehicles. All these datasets have one feature channel (the traffic speed or the traffic flow), *i.e.,* $C = 1$.

Construction of the traffic network. For traffic speed datasets, we follow the procedure of DCRNN [21]. We compute the pairwise road network distances between sensors and build the adjacency matrix using thresholded Gaussian kernel [30]. For traffic flow datasets, we use the traffic network provided by ASTGCN [11]. They remove many redundant detectors to ensure the distance between any adjacent detectors is longer than 3.5 miles to obtain a lightweight traffic network. The following gives more detailed description of the four datasets:

- **METR-LA** is a public traffic speed dataset collected from loop-detectors located on the LA County road network [15]. Specifically, METR-LA contains data of 207 sensors over a period of 4 months from Mar 1st 2012 to Jun 30th 2012 [21]. The traffic information is recorded at the rate of every 5 minutes, and the total number of time slices is 34,272.
- **PEMS-BAY** is a public traffic speed dataset collected from California Transportation Agencies (CalTrans) Performance Measurement System (PeMS) [5]. Specifically, PEMS-BAY contains data of 325 sensors in the Bay Area over a period of 6 months from Jan 1st 2017 to May 31th 2017 [21]. The traffic information is recorded at the rate of every 5 minutes, and the total number of time slices is 52,116.
- **PEMS04** is a public traffic flow dataset collected from CalTrans PeMS [5]. Specifically, PEMS04 contains data of 307 sensors in the District04 over a period of 2 months from Jan 1st 2018 to Feb 28th 2018 [11]. The traffic information is recorded at the rate of every 5 minutes, and the total number of time slices is 16,992.
- **PEMS08** is a public traffic flow dataset collected from CalTrans PeMS [5]. Specifically, PEMS08 contains data of 170 sensors in the District08 over a period of 2 months from July 1st 2018 to Aug 31th 2018 [11]. The traffic information is recorded at the rate of every 5 minutes, and the total number of time slices is 17,833.

Baselines. We select a wealth of baselines that have official public code, including the traditional methods and the typical deep learning methods, as well as the very recent state-of-the-art works.

- **HA**: Historical Average model, which models traffic flows as a periodic process and uses weighted averages from previous periods as predictions for future periods.
- **VAR**: Vector Auto-Regression [22, 23] assumes that the passed time series is stationary and estimates the relationship between the time series and their lag value. [37]
- **SVR**: Support Vector Regression (SVR) uses linear support vector machine for classical time series regression task.
- **FC-LSTM** [32]: Long Short-Term Memory network with fully connected hidden units is a well-known network architecture that is powerful in capturing sequential dependency.
- **DCRNN** [21]: Diffusion Convolutional Recurrent Neural Network [21] models the traffic flow as a diffusion process. It replaces the fully connected layer in GRU [7] by diffusion convolutional layer to form a new Diffusion Convolutional Gated Recurrent Unit (DCGRU).
- **Graph WaveNet** [41]: Graph WaveNet stacks Gated TCN and GCN layer by layer to jointly capture the spatial and temporal dependencies.
- **ASTGCN** [11]: ASTGCN combines the spatial-temporal attention mechanism to capture the dynamic spatial-temporal characteristics of traffic data simultaneously.
- **STSGCN** [31]: STSGCN is proposed to effectively capture the localized spatial-temporal correlations and consider the heterogeneity in spatial-temporal data.
- **GMAN** [51]: GMAN is an attention-based model which stacks spatial, temporal and transform attentions.
- **MTGNN** [40]: MTGNN extends Graph WaveNet through the mix-hop propagation layer in the spatial module, the dilated inception layer in the temporal module, and a more delicate graph learning layer.
- **DGCRN** [20]: DGCRN models the dynamic graph and designs a novel Dynamic Graph Convolutional Recurrent Module (DGCRM) to capture the spatial-temporal pattern in a seq2seq architecture.

We use the default settings as described in baseline papers. We evaluate the performances of all baselines by three commonly used metrics in traffic forecasting, including Mean Absolute Error (MAE), Root Mean Squared Error (RMSE) and Mean Absolute Percentage Error (MAPE). The formulas are as follows:

$$\begin{aligned} \mathrm{MAE}(x,\hat{x}) &= \frac{1}{|\Omega|}\sum_{i\in\Omega}|x_i-\hat{x}_i|, \\ \mathrm{RMSE}(x,\hat{x}) &= \sqrt{\frac{1}{|\Omega|}\sum_{i\in\Omega}(x_i-\hat{x}_i)^2}, \\ \mathrm{MAPE}(x,\hat{x}) &= \frac{1}{|\Omega|}\sum_{i\in\Omega}\frac{|x_i-\hat{x}_i|}{x_i}, \end{aligned} \tag{17}$$

where MAE metric reflects the prediction accuracy [20], RMSE is more sensitive to abnormal values, and MAPE can eliminate the influence of data units to some extent. x_i denotes the i-th ground truth, $\hat{x}_i$ represents the i-th predicted values, and Ω is the indices of observed samples, where $|\Omega| = T_f = 12$ in our experiments.

Implementation. The proposed model is implemented by Pytorch 1.9.1 on NVIDIA 3090 GPU. We use Adam as our optimizer and set the learning rate to 0.001. The embedding size of nodes and time slots is set to 12. The other hidden dimension d in this paper is set to 32. The spatial kernel size is set to 2, and the temporal kernel size is set to 3 for all datasets. The batch size is set to 32. We employ early stopping to avoid overfitting. We perform significance test (t-test with p-value < 0.05) over all the experimental results. For any other more details, readers could refer to our public code repository.

6.2 The Performance of D^{2}STGNN

6.2.1 Settings. For a fair comparison, we follow the dataset division in previous works. For METR-LA and PEMS-BAY, we use about 70% of data for training, 20% of data for testing, and the remaining 10% for validation [20, 21, 40, 41]. For PEMS04 and PEMS08, we use about 60% of data for training, 20% of data for testing, and the remaining 20% for validation [11, 12, 31]. We generate sequence samples through a sliding window with a width of 24 (2 hours), where the first 12 time steps are used as input, and the remaining 12 time steps are used as ground truth. We compare the performance of 15 minutes (horizon 3), 30 minutes (horizon 6), and 1 hour (horizon 12) ahead forecasting on the MAE, RMSE, and MAPE metrics.

6.2.2 Results. As shown in Table 3, D^{2}STGNN consistently achieves the best performance in all horizons in all datasets, which indicates the effectiveness of our model. Traditional methods such as HA, SVR perform worst because of their strong assumption about the data, *e.g.,* stationary or linear. FC-LSTM, a classic recurrent neural network for sequential data, can not perform well since it only considers temporal features, but ignores the spatial impact in traffic data and, which is crucial in traffic forecasting. VAR takes both spatial and temporal information into consideration, thus it achieves better performance. However, VAR cannot capture strong nonlinear and dynamic spatial-temporal correlations. Recently proposed spatial-temporal models overcome these shortcomings and make considerable progress. DCRNN and Graph WaveNet are two typical spatial-temporal coupling models among them. Graph WaveNet combines GNN and Gated TCN to form a spatial-temporal layer while DCRNN replaces the fully connected layer in GRU by diffusion convolution to get a diffusion convolutional GRU. Even if compared with many of the latest works, such as ASTGCN and STSGCN, their performance is still very promising. This may be due to their refined data assumptions and reasonable model architecture. MTGNN replaces the GNN and Gated TCN in Graph WaveNet with mix-hop propagation layer [1] and dilated inception layer, and proposes the learning of latent adjacency matrix to seek further improvement. GMAN performs better in long-term prediction thanks to the attention mechanism's powerful ability to capture long-term dependency. Based on the DCRNN architecture, DGCRN captures the dynamic characteristics of the spatial topology and achieves better performance than other baselines. Our model still outperforms DGCRN. We conjecture the key reason lies in the decoupled ST framework. In a nutshell, the results in Table 3 validate the superiority of D^{2}STGNN.

Note that the final performance is affected by many aspects: the modeling of temporal and spatial dependencies and dynamic spatial topology. Therefore, although Table 3 has shown the superiority of the D^{2}STGNN model, it is not enough to evaluate the effectiveness of the proposed decoupled spatial-temporal framework.

6.3 Effectiveness of the Decoupled Framework and the Spatial-Temporal Model

In this subsection, we conduct experiments to verify the effectiveness of the decoupled spatial-temporal framework as well as the diffusion and inherent model. As mentioned before, we remove the dynamic spatial dependency learning module in all methods, *e.g.,* dynamic graph learner in our model, for a fair comparison.

On the one hand, we need to compare D^{2}STGNN with its variant without the DSTF, named the coupled version of D^{2}STGNN, where the two hidden time series remain coupled like in other STGNNs. On the other hand, we also want to compare the coupled version of D^{2}STGNN with the other STGNNs to test the effectiveness of the diffusion and inherent model. To this end, we first replace the dynamic graph in D^{2}STGNN with the pre-defined static graph to get D^{2}STGNN†. Based on it, we consider D^{2}STGNN‡, which removes the DSTF by removing the estimation gate and residual decomposition, and connects the diffusion model and inherent model directly. We select the two most representative baselines, Graph WaveNet (GWNet) and DGCRN. For a fair comparison, the dynamic adjacency matrix in DGCRN is also removed, named DGCRN†.

The result is shown in Table 4. We have the following findings. (i) D^{2}STGNN† significantly outperforms D^{2}STGNN‡, which shows that the DSTF is crucial in our model. (ii) The coupled version D^{2}STGNN‡ can also perform better than baselines, which indicates the effectiveness of our diffusion and inherent model. However, the D^{2}STGNN‡ has only limited advantages compared with other baselines, which again shows the importance of decoupling the two hidden time series in the original traffic data.

6.4 Efficiency

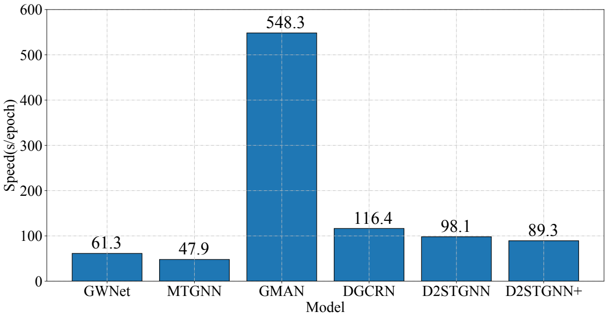


Figure 6: Average training time per epoch.

In this part, we compare the efficiency of D^{2}STGNN with other methods based on the METR-LA dataset. For a more intuitive and effective comparison, we compare the average training time required for each epoch of these models. Specifically, we compare the speed of D^{2}STGNN, D^{2}STGNN† (without dynamic graph learning), DGCRN, GMAN, MTGNN, and Graph WaveNet. All models are running on Intel(R) Xeon(R) Gold 5217 CPU @ 3.00GHz, 128G RAM computing server, equipped with RTX 3090 graphics card. The batch size is uniformly set to 32.

As shown in Figure 6, D^{2}STGNN does not increase the computational burden too much compared to other baselines. In addition, it achieves both better performance and higher efficiency in the same time than other state-of-the-art baselines like GMAN and DGCRN. This is mainly due to the fact that the decoupled framework (*i.e.,*estimation gate and residual decomposition mechanism) focuses on developing a more reasonable structures connecting diffusion and inherent models to improve the model's capabilities,

Table 3: Traffic forecasting on the METR-LA, PEMS-BAY, PEMS04,and PEMS08 datasets. Numbers marked with * indicate that the improvement is statistically significant compared with the best baseline (t-test with p-value< 0.05).

Datasets	Methods	Horizon 3			Horizon 6			Horizon 12		
		MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
METR-LA	HA	4.79	10.00	11.70%	5.47	11.45	13.50%	6.99	13.89	17.54%
	VAR	4.42	7.80	13.00%	5.41	9.13	12.70%	6.52	10.11	15.80%
	SVR	3.39	8.45	9.30%	5.05	10.87	12.10%	6.72	13.76	16.70%
	FC-LSTM	3.44	6.30	9.60%	3.77	7.23	10.09%	4.37	8.69	14.00%
	DCRNN	2.77	5.38	7.30%	3.15	6.45	8.80%	3.60	7.60	10.50%
	STGCN	2.88	5.74	7.62%	3.47	7.24	9.57%	4.59	9.40	12.70%
	Graph WaveNet	2.69	5.15	6.90%	3.07	6.22	8.37%	3.53	7.37	10.01%
	ASTGCN	4.86	9.27	9.21%	5.43	10.61	10.13%	6.51	12.52	11.64%
	STSGCN	3.31	7.62	8.06%	4.13	9.77	10.29%	5.06	11.66	12.91%
	MTGNN	2.69	5.18	6.88%	3.05	6.17	8.19%	3.49	7.23	9.87%
	GMAN	2.80	5.55	7.41%	3.12	6.49	8.73%	3.44	7.35	10.07%
	DGCRN	2.62	5.01	6.63%	2.99	6.05	8.02%	3.44	7.19	9.73%
	D^{2}STGNN	**2.56***	**4.88***	**6.48%***	**2.90***	**5.89***	**7.78%***	**3.35***	**7.03***	**9.40%***
PEMS-BAY	HA	1.89	4.30	4.16%	2.50	5.82	5.62%	3.31	7.54	7.65%
	VAR	1.74	3.16	3.60%	2.32	4.25	5.00%	2.93	5.44	6.50%
	SVR	1.85	3.59	3.80%	2.48	5.18	5.50%	3.28	7.08	8.00%
	FC-LSTM	2.05	4.19	4.80%	2.20	4.55	5.20%	2.37	4.96	5.70%
	DCRNN	1.38	2.95	2.90%	1.74	3.97	3.90%	2.07	4.74	4.90%
	STGCN	1.36	2.96	2.90%	1.81	4.27	4.17%	2.49	5.69	5.79%
	Graph WaveNet	1.30	2.74	2.73%	1.63	3.70	3.67%	1.95	4.52	4.63%
	ASTGCN	1.52	3.13	3.22%	2.01	4.27	4.48%	2.61	5.42	6.00%
	STSGCN	1.44	3.01	3.04%	1.83	4.18	4.17%	2.26	5.21	5.40%
	MTGNN	1.32	2.79	2.77%	1.65	3.74	3.69%	1.94	4.49	4.53%
	GMAN	1.34	2.91	2.86%	1.63	3.76	3.68%	1.86	4.32	**4.37%**
	DGCRN	1.28	2.69	2.66%	1.59	3.63	3.55%	1.89	4.42	4.43%
	D^{2}STGNN	**1.24***	**2.60***	**2.58%***	**1.55***	**3.52***	**3.49%***	**1.85***	**4.30***	**4.37%**
PEMS04	HA	28.92	42.69	20.31%	33.73	49.37	24.01%	46.97	67.43	35.11%
	VAR	21.94	34.30	16.42%	23.72	36.58	18.02%	26.76	40.28	20.94%
	SVR	22.52	35.30	14.71%	27.63	42.23	18.29%	37.86	56.01	26.72%
	FC-LSTM	21.42	33.37	15.32%	25.83	39.10	20.35%	36.41	50.73	29.92%
	DCRNN	20.34	31.94	13.65%	23.21	36.15	15.70%	29.24	44.81	20.09%
	STGCN	19.35	30.76	12.81%	21.85	34.43	14.13%	26.97	41.11	16.84%
	Graph WaveNet	18.15	29.24	12.27%	19.12	30.62	13.28%	20.69	33.02	14.11%
	ASTGCN	20.15	31.43	14.03%	22.09	34.34	15.47%	26.03	40.02	19.17%
	STSGCN	19.41	30.69	12.82%	21.83	34.33	14.54%	26.27	40.11	14.71%
	MTGNN	18.22	30.13	12.47%	19.27	32.21	13.09%	20.93	34.49	14.02%
	GMAN	18.28	29.32	12.35%	18.75	30.77	12.96%	19.95	**30.21**	12.97%
	DGCRN	18.27	28.97	12.36%	19.39	30.86	13.42%	21.09	33.59	14.94%
	D^{2}STGNN	**17.44***	**28.64***	**11.64%***	**18.28***	**30.10***	**12.10%***	**19.55***	31.99	**12.82%***
PEMS08	HA	23.52	34.96	14.72%	27.67	40.89	17.37%	39.28	56.74	25.17%
	VAR	19.52	29.73	12.54%	22.25	33.30	14.23%	26.17	38.97	17.32%
	SVR	17.93	27.69	10.95%	22.41	34.53	13.97%	32.11	47.03	20.99%
	FC-LSTM	17.38	26.27	12.63%	21.22	31.97	17.32%	30.69	43.96	25.72%
	DCRNN	15.64	25.48	10.04%	17.88	27.63	11.38%	22.51	34.21	14.17%
	STGCN	15.30	25.03	9.88%	17.69	27.27	11.03%	25.46	33.71	13.34%
	Graph WaveNet	14.02	22.76	8.95%	15.24	24.22	9.57%	16.67	26.77	10.86%
	ASTGCN	16.48	25.09	11.03%	18.66	28.17	12.23%	22.83	33.68	15.24%
	STSGCN	15.45	24.39	10.22%	16.93	26.53	10.84%	19.50	30.43	12.27%
	MTGNN	14.24	22.43	9.02%	15.30	24.32	9.58%	16.85	26.93	10.57%
	GMAN	13.80	22.88	9.41%	14.62	24.02	9.57%	15.72	**25.96**	10.56%
	DGCRN	13.89	22.07	9.19%	14.92	23.99	9.85%	16.73	26.88	10.84%
	D^{2}STGNN	**13.14***	**21.42***	**8.55%***	**14.21***	**23.65***	**9.12%***	**15.69***	26.41	**10.17%***

Table 4: Comparison of decoupled and coupled ST Framework. H denotes horizon. Numbers marked with * indicate that the improvement is statistically significant compared with the best baseline (t-test with p-value< 0.05).

			GWNet	DGCRN†	D^{2}STGNN‡	D^{2}STGNN†
METR-LA	H 3	MAE	2.69	2.71	2.66	**2.59***
		RMSE	5.15	5.19	5.10	**4.99***
		MAPE	6.90%	7.04%	6.80%	**6.69%***
	H 6	MAE	3.07	3.12	3.04	**2.93***
		RMSE	6.22	6.31	6.13	**5.97***
		MAPE	8.37%	8.60%	8.24%	**7.99%***
	H 12	MAE	3.53	3.64	3.51	**3.38***
		RMSE	7.37	7.59	7.27	**7.07***
		MAPE	10.01%	10.62%	10.02%	**9.63%***
PEMS-BAY	H 3	MAE	1.30	1.32	1.31	**1.25***
		RMSE	2.74	2.78	2.74	**2.64***
		MAPE	2.73%	2.78%	2.76%	**2.64%***
	H 6	MAE	1,63	1.66	1.63	**1.55***
		RMSE	3.70	3.78	3.66	**3.56***
		MAPE	3.67%	3.76%	3.66%	**3.58%***
	H 12	MAE	1.95	1.99	1.94	**1.85***
		RMSE	4.52	4.60	4.50	**4.33***
		MAPE	4.63%	4.73%	4.59%	**4.43%***
PEMS04	H 3	MAE	18.15	18.97	18.94	**17.55***
		RMSE	29.24	30.01	29.38	**28.70***
		MAPE	12.27%	13.38%	13.58%	**11.78%***
	H 6	MAE	19.12	20.30	20.14	**18.38***
		RMSE	30.62	31.78	31.54	**30.15***
		MAPE	13.28%	14.48%	15.11%	**12.26%***
	H 12	MAE	20.69	22.95	22.57	**19.59***
		RMSE	33.02	35.15	34.33	**32.04***
		MAPE	14.11%	16.97%	17.16%	**12.95%***
PEMS08	H 3	MAE	14.02	14.54	14.49	**13.28***
		RMSE	22.76	22.62	22.35	**21.56***
		MAPE	8.95%	9.37%	10.14%	**8.53%***
	H 6	MAE	15.24	15.64	15.69	**14.26***
		RMSE	24.22	24.52	24.37	**23.49***
		MAPE	9.57%	10.03%	10.41%	**9.15%***
	H 12	MAE	16.67	17.80	18.01	**15.65***
		RMSE	26.77	27.92	27.53	**25.78***
		MAPE	10.86%	11.71%	11.81%	**10.10%***

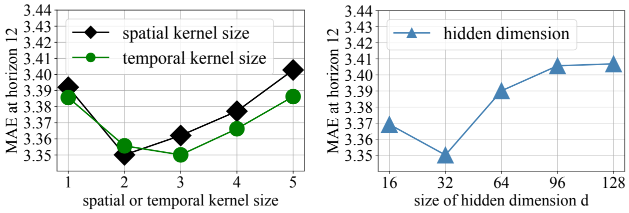
rather than increasing the complexity of the diffusion or inherent models. Graph WaveNet and MTGNN are the most efficient, thanks to their lightweight and easily parallelized models. But their performance is worse than other models.

6.5 Ablation Study

In this part, we will conduct ablation studies from three aspects to verify our work: the architecture of decoupled spatial-temporal framework, the important components, and the training strategy. Firstly, we design four variants of our decoupled spatial-temporal framework. *Switch* places inherent block before diffusion block in each layer to verify that whether they are interchangeable. *W/o gate* removes the estimation gate in the decouple block, while *w/o res* removes the residual links. *W/o decouple* removes the estimation gate and residual decomposition simultaneously (*i.e.,* the D^{2}STGNN‡ in Table 4). Secondly, we test the effectiveness of four important components. *W/o dg* replaces the dynamic graph with a pre-defined static graph (*i.e.,* the D^{2}STGNN† in Table 4). *W/o apt* removes the self-adaptive transition matrix in the diffusion model. *W/o gru* removes the GRU layer in the inherent model, while *w/o msa* removes the multi-head self-attention layer. Thirdly, we design two variants to test the effectiveness of the training strategy: *w/o ar* removes the auto-regression strategy in the forecast branch and directly applies a regression layer on the hidden state to forecast multi-steps at once, *w/o cl* removes the curriculum learning.

The result is shown in Table 5. On the architecture aspect, switching the diffusion and inherent model does not make a significant difference. The results of *w/o gate* and *w/o res* suggest that the estimation gate and residual decomposition mechanism are both important for decoupling. The results of *w/o decouple* show that decoupling the two hidden signals is crucial for accurate traffic forecasting. On the important components aspect, the dynamic graph learning model provides consistent performance improvements compared with the pre-defined static graph. The results of *w/o gru* and *w/o msa* in the inherent model show that both short- and long-term dependencies are crucial for accurate traffic forecasting. On the training strategy aspect, the result of *w/o ar* indicates that the auto-regressive forecast strategy is more suitable for our model, and the result of *w/o cl* shows that correct training strategy can help the model to converge better.

6.6 Parameter Sensitivity



(a) Impact of kernel size k_t and k_s

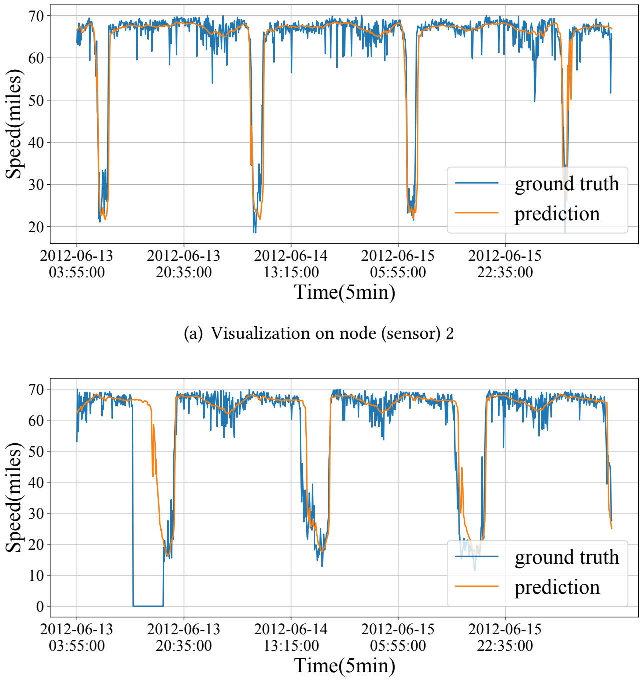
(b) Impact of hidden dimension d

Figure 7: Parameter sensitivity of D^{2}STGNN.

In this section, we conduct experiments to analyze the impacts of three critical hyper-parameters: spatial kernel size k_s, temporal kernel size k_t, and hidden dimension d. In Figure 7, we present the traffic forecasting result on METR-LA dataset with different parameters. The effect of spatial kernel size k_s and temporal kernel size k_t is shown in Figure 7(a). We set the range of k_s and k_t from 1 to 5 individually. The experimental results verify the spatial-temporal localized characteristics of diffusion process. In addition, the effect of hidden dimension d is shown in Figure 7(b), which shows that a smaller dimension is insufficient to encode the spatial and temporal information, while the larger dimension may introduce overfitting.

Table 5: Ablation study on METR-LA.

Variants	Horizon 3			Horizon 6			Horizon 12		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
D^{2}STGNN	**2.56**	**4.88**	**6.48%**	**2.90**	**5.89**	**7.78%**	**3.35**	**7.03**	**9.40%**
switch	**2.56**	4.90	6.50%	2.91	5.92	7.85 %	**3.35**	7.05	9.47%
w/o gate	2.60	4.98	6.63%	2.96	6.01	8.07 %	3.44	7.16	9.78%
w/o res	2.60	4.96	6.84%	2.93	5.95	8.21 %	3.37	7.10	9.80%
w/o decouple	2.66	5.10	6.80%	3.04	6.13	8.24 %	3.51	7.27	10.02%
w/o dg	2.59	4.99	6.69%	2.93	5.97	7.99 %	3.38	7.07	9.63%
w/o apt	2.58	4.92	6.51%	2.93	5.92	7.80 %	3.40	7.10	9.43%
w/o gru	2.59	4.93	6.66%	2.94	6.02	7.98 %	3.38	7.07	9.66%
w/o msa	2.59	4.95	6.60%	2.93	5.96	7.99 %	3.37	7.09	9.67%
w/o ar	2.59	4.98	6.61%	2.94	5.96	7.95 %	3.39	7.09	9.64%
w/o cl	2.62	5.01	6.70%	2.96	6.02	8.05 %	3.38	7.08	9.63%



(a) Visualization on node (sensor) 2

(b) Visualization on node (sensor) 111

Figure 8: Visualization of prediction results on METR-LA.

6.7 Visualization

In order to further intuitively understand and evaluate our model, in this section, we visualize the prediction of our model and the ground-truth. Due to space limitations, we randomly selected two nodes and displayed their data from June 13th 2012 to June 16th 2012 (located in the test dataset). The forecasting results on node 2 and node 111 are shown in Figure 8. It is obvious that the patterns of the two selected nodes are different. For example, there is often traffic congestion during the morning peak hour at sensor 2, while sensor 111 often records traffic congestion during the evening peak hours. The results indicate that our model can capture these unique patterns for different nodes. Furthermore, it can be seen that the model is very good at capturing the inherent patterns of time series while avoiding overfitting the noise. For example, sensor 111 apparently failed in the afternoon of June 13, 2012, where the records suddenly were zero. However, our model does not forcefully fit these noises and correctly predicted the traffic congestion. Furthermore, as shown in Figure 8, the model achieved very impressive prediction accuracy on the whole, while the prediction in some local details may not be accurate due to large random noise.

7 CONCLUSION

In this paper, we first propose to decouple the diffusion signal and inherent signal from traffic data by a novel **D**ecoupled **S**patial **T**emporal **F**ramework (DSTF). This enables more precise modeling of the different parts of traffic data, thus promising to improve prediction accuracy. Based on the novel DSTF, **D**ecoupled **D**ynamic **S**patial-**T**emporal **G**raph **N**eural **N**etwork(D^{2}STGNN) is proposed by carefully designing the diffusion and inherent model as well as the dynamic graph learning model according to the characteristics of diffusion signals and inherent signals. Specifically, a spatial-temporal localized convolution is designed to model *the hidden diffusion time series*. The recurrent neural network and self-attention mechanism are jointly used to model the *hidden inherent time series*. Furthermore, the dynamic graph learning module comprehensively exploits different information to adjust the road network-based spatial dependency by learning the latent correlations between time series based on the self-attention mechanism. Extensive experiments on four real-world datasets show that our proposal is able to consistently and significantly outperform all baselines.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant No. 61902376, No. 61902382, and No. 61602197, in part by CCF-AFSG Research Fund under Grant No. RF20210005, and in part by the fund of Joint Laboratory of HUST and Pingan Property & Casualty Research (HPL). In addition, Zhao Zhang is supported by the China Postdoctoral Science Foundation under Grant No. 2021M703273.

REFERENCES

[1] Sami Abu-El-Haija, Bryan Perozzi, Amol Kapoor, Nazanin Alipourfard, Kristina Lerman, Hrayr Harutyunyan, Greg Ver Steeg, and Aram Galstyan. 2019. Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing. In *international conference on machine learning*. PMLR, 21–29.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral Networks and Locally Connected Networks on Graphs. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. http://arxiv.org/abs/1312.6203

[3] Ennio Cascetta. 2013. *Transportation systems engineering: theory and methods*. Vol. 49. Springer Science & Business Media.

[4] Suresh Chavhan and Pallapa Venkataram. 2020. Prediction based traffic management in a metropolitan area. *Journal of traffic and transportation engineering (English edition)* 7, 4 (2020), 447–466.

[5] Chao Chen, Karl Petty, Alexander Skabardonis, Pravin Varaiya, and Zhanfeng Jia. 2001. Freeway performance measurement system: mining loop detector data. *Transportation Research Record* 1748, 1 (2001), 96–102.

[6] Xu Chen, Yuanxing Zhang, Lun Du, Zheng Fang, Yi Ren, Kaigui Bian, and Kunqing Xie. 2020. Tssrgcn: Temporal spectral spatial retrieval graph convolutional network for traffic flow forecasting. In *2020 IEEE International Conference on Data Mining (ICDM)*. IEEE, 954–959.

[7] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder-Decoder Approaches. In *SSST@EMNLP*. 103–111.

[8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. 3837–3845.

[9] Ziquan Fang, Lu Pan, Lu Chen, Yuntao Du, and Yunjun Gao. 2021. MDTP: A Multi-source Deep Traffic Prediction Framework over Spatio-Temporal Trajectory Data. *Proc. VLDB Endow.* 14, 8 (2021), 1289–1297.

[10] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. 2011. Deep sparse rectifier neural networks. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 315–323.

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

[12] Shengnan Guo, Youfang Lin, Huaiyu Wan, Xiucheng Li, and Gao Cong. 2021. Learning dynamics and heterogeneity of spatial-temporal graph data for traffic forecasting. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[13] Liangzhe Han, Bowen Du, Leilei Sun, Yanjie Fu, Yisheng Lv, and Hui Xiong. 2021. Dynamic and Multi-faceted Spatio-temporal Deep Learning for Traffic Speed Forecasting. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 547–555.

[14] Xiaolin Han, Tobias Grubenmann, Reynold Cheng, Sze Chun Wong, Xiaodong Li, and Wenya Sun. 2020. Traffic incident detection: A trajectory-based approach. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. IEEE, 1866–1869.

[15] Hosagrahar V Jagadish, Johannes Gehrke, Alexandros Labrinidis, Yannis Papakonstantinou, Jignesh M Patel, Raghu Ramakrishnan, and Cyrus Shahabi. 2014. Big data and its technical challenges. *Commun. ACM* 57, 7 (2014), 86–94.

[16] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

[17] Genshiro Kitagawa and Will Gersch. 1984. A smoothness priors–state space modeling of time series with trend and seasonality. *J. Amer. Statist. Assoc.* 79, 386 (1984), 378–389.

[18] S Vasantha Kumar and Lelitha Vanajakshi. 2015. Short-term traffic flow prediction using seasonal ARIMA model with limited input data. *European Transport Research Review* 7, 3 (2015), 1–9.

[19] Hyunwook Lee, Cheonbok Park, Seungmin Jin, Hyeshin Chu, Jaegul Choo, and Sungahn Ko. 2021. An Empirical Experiment on Deep Learning Models for Predicting Traffic Data. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 1817–1822.

[20] Fuxian Li, Jie Feng, Huan Yan, Guangyin Jin, Depeng Jin, and Yong Li. 2021. Dynamic Graph Convolutional Recurrent Network for Traffic Prediction: Benchmark and Solution. *arXiv preprint arXiv:2104.14917* (2021).

[21] Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

[22] Zheng Lu, Chen Zhou, Jing Wu, Hao Jiang, and Songyue Cui. 2016. Integrating granger causality and vector auto-regression for traffic prediction of large-scale WLANs. *KSII Transactions on Internet and Information Systems (TIIS)* 10, 1 (2016), 136–151.

[23] Helmut Lütkepohl. 2005. *New introduction to multiple time series analysis*. Springer Science & Business Media.

[24] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2014. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* 16, 2 (2014), 865–873.

[25] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2019. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*.

[26] Zheyi Pan, Yuxuan Liang, Weifeng Wang, Yong Yu, Yu Zheng, and Junbo Zhang. 2019. Urban traffic prediction from spatio-temporal data using deep meta learning. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1720–1730.

[27] Cheonbok Park, Chunggi Lee, Hyojin Bahng, Yunwon Tae, Seungmin Jin, Kihwan Kim, Sungahn Ko, and Jaegul Choo. 2020. ST-GRAT: A novel spatio-temporal graph attention networks for accurately forecasting dynamically changing road speed. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 1215–1224.

[28] Tangwen Qian, Fei Wang, Yongjun Xu, Yu Jiang, Tao Sun, and Yong Yu. 2020. CABIN: a novel cooperative attention based location prediction network using internal-external trajectory dependencies. In *International Conference on Artificial Neural Networks*. Springer, 521–532.

[29] Zezhi Shao, Zhao Zhang, Fei Wang, and Yongjun Xu. 2022. Pre-training Enhanced Spatial-temporal Graph Neural Network for Multivariate Time Series Forecasting. *arXiv e-prints* (2022), arXiv–2206.

[30] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. 2013. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine* 30, 3 (2013), 83–98.

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

[32] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[33] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[34] Luan Tran, Min Y Mun, Matthew Lim, Jonah Yamato, Nathan Huh, and Cyrus Shahabi. 2020. DeepTRANS: a deep learning system for public bus travel time estimation using traffic forecasting. *Proceedings of the VLDB Endowment* 13, 12 (2020), 2957–2960.

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[36] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

[37] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of The Web Conference 2020*. 1082–1092.

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

[39] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. 2019. Simplifying graph convolutional networks. In *International conference on machine learning*. PMLR, 6861–6871.

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

[41] Zonghan Wu, Shirui Pan, Guodong Long, Jing Jiang, and Chengqi Zhang. 2019. Graph WaveNet for Deep Spatial-Temporal Graph Modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. 1907–1913.

[42] Yongjun Xu, Xin Liu, Xin Cao, Changping Huang, Enke Liu, Sen Qian, Xingchen Liu, Yanjun Wu, Fengliang Dong, Cheng-Wei Qiu, et al. 2021. Artificial intelligence: A powerful paradigm for scientific research. *The Innovation* 2, 4 (2021).

[43] Huaxiu Yao, Fei Wu, Jintao Ke, Xianfeng Tang, Yitian Jia, Siyu Lu, Pinghua Gong, Jieping Ye, and Zhenhui Li. 2018. Deep Multi-View Spatial-Temporal Network for Taxi Demand Prediction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*. 2588–2595.

[44] Junchen Ye, Leilei Sun, Bowen Du, Yanjie Fu, and Hui Xiong. 2021. Coupled Layer-wise Graph Convolution for Transportation Demand Prediction. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021.* 4617–4625.

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

[46] Fisher Yu and Vladlen Koltun. 2016. Multi-Scale Context Aggregation by Dilated Convolutions. In *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings.*

[47] Haitao Yuan, Guoliang Li, Zhifeng Bao, and Ling Feng. 2021. An effective joint prediction model for travel demands and traffic flows. In *2021 IEEE 37th International Conference on Data Engineering (ICDE).* IEEE, 348–359.

[48] Zhuoning Yuan, Xun Zhou, and Tianbao Yang. 2018. Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining.* 984–992.

[49] Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. 2018. GaAN: Gated Attention Networks for Learning on Large and Spatiotemporal Graphs. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence, UAI 2018, Monterey, California, USA, August 6-10, 2018.* 339–349.

[50] Qi Zhang, Jianlong Chang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. 2020. Spatio-temporal graph structure learning for traffic forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 1177–1185.

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

[52] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. 2014. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)* 5, 3 (2014), 1–55.

[53] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021.* 11106–11115.