Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

MltAuxTSPP: a unified benchmark for deep learning-based traffic state prediction with multi-source auxiliary data*#

Yusong ZHOU†1,2, Xiaoyu JIANG†1,2, Shu SUN†1,2,
Xinmin ZHANG†‡1,2, Yuanqiu MO[3], Zhihuan SONG[1,2]

[1]*College of Control Science and Engineering, Zhejiang University, Hangzhou 310027, China*

[2]*State Key Laboratory of Industrial Control Technology, Zhejiang University, Hangzhou 310027, China*

[3]*School of Mathematics, Southeast University, Nanjing 211189, China*

†E-mail: zhoushichan@zju.edu.cn; jiangxiaoyu@zju.edu.cn; shusun@zju.edu.cn; xinminzhang@zju.edu.cn

Received Mar. 17, 2025; Revision accepted Aug. 17, 2025; Crosschecked Sept. 18, 2025

Abstract: Deep learning has empowered traffic prediction models to integrate diverse auxiliary data sources, such as weather and temporal features, for enhanced forecasting accuracy. However, existing approaches often suffer from limited generality and scalability, and the field lacks a unified benchmark for fair model comparison. This absence hinders consistent performance evaluation, slows the development of robust and adaptable models, and makes it challenging to quantify the incremental benefits of different auxiliary data sources. To address these issues, we present MltAuxTSPP, a unified benchmark framework for deep learning-based traffic state prediction with multi-source auxiliary data. The framework features a standardized data container and a fusion embedding module, enabling consistent utilization of heterogeneous data and improving scalability. It produces unified hidden representations that can be seamlessly adopted by various downstream models, ensuring fair and reproducible comparisons under identical conditions. Extensive experiments on real-world datasets demonstrate that MltAuxTSPP effectively leverages weather and temporal features to improve long-term forecast performance and offers a practical and reproducible foundation for advancing research in traffic state prediction.

Key words: Traffic prediction; Benchmark platform; Deep learning; Multi-source auxiliary data
https://doi.org/10.1631/FITEE.2500169 **CLC number:** TP18

1 Introduction

Accurate traffic state prediction is a cornerstone of modern intelligent transportation systems (ITSs) and is essential for applications ranging from traffic management to autonomous driving. Although deep learning models have excelled at capturing complex spatiotemporal dependencies from traffic data (Do et al., 2019; Zhu et al., 2019; Lee et al., 2021), their performance is often constrained by relying solely on traffic-related data, such as the fixed position sensor data, trajectory data, network infrastructure data, and trip records data (Nagy and Simon, 2018; Shaygan et al., 2022).

Real-world traffic is significantly influenced by external factors, such as predictable temporal patterns (e.g., weekday rush hours vs. weekends) and adverse weather conditions (Essien et al., 2018; Taghipour et al., 2020). Consequently, recent studies have begun to integrate such auxiliary data to enhance prediction accuracy (Taghipour et al., 2020; Essien et al., 2021). However, the integration of multi-source data remains fraught with challenges.

‡ Corresponding author

* Project supported by the National Science and Technology Major Project of China (No. 2022ZD0120003)

Electronic supplementary materials: The online version of this article (https://doi.org/10.1631/FITEE.2500169) contains supplementary materials, which are available to authorized users

ORCID: Yusong ZHOU, https://orcid.org/0009-0003-4475-623X; Xinmin ZHANG, https://orcid.org/0000-0002-4761-3969

Current approaches often handle a limited set of auxiliary variables, which typically lack spatial dimensions. Furthermore, these methods generally lack the generality and scalability required to accommodate diverse and growing data sources. Crucially, the field is hampered by the absence of a unified benchmark platform for systematically evaluating and comparing models under multi-source data conditions.

These limitations collectively pose significant obstacles to advancing the field. The lack of generality hinders the application of existing methods to new datasets or information streams. Poor scalability makes it challenging to handle the increasing volume and variety of data. Crucially, the absence of a unified benchmark leads to fragmented research, impedes objective comparison of models, and ultimately slows ITS innovation.

To overcome these obstacles, we introduce MltAuxTSPP, a novel unified benchmark framework for deep learning-based traffic state prediction using multi-source auxiliary data. At its core, MltAuxTSPP features a unified data container (UDC) and a versatile fusion embedding module (Fig. 1). This design addresses critical challenges in data handling and scalability, enabling seamless integration with various existing spatiotemporal models. This allows researchers to fairly evaluate and compare different model architectures on a standardized basis. The main contributions of this work are as follows:

1. We design and propose MltAuxTSPP, a general and scalable framework for enhancing traffic prediction with multi-source data. Its fusion module standardizes the integration of diverse data types for various downstream models.

2. We construct and release two new multi-source traffic forecasting datasets by augmenting the widely used METR-LA and PEMS-BAY benchmarks with extensive weather and temporal features, facilitating standardized evaluation. Note that the METR-LA data are listed in the supplementary materials.

3. We conduct extensive experiments to systematically evaluate the impact of auxiliary data on state-of-the-art models and provide novel insights into the significant benefits of multi-source data for long-term forecasting.

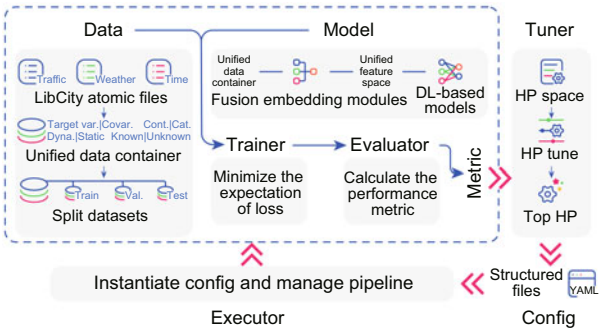

Fig. 1 The benchmarking system in MltAuxTSPP. DL: deep learning; HP: hyperparameter; var.: variable; covar.: covariance; cont.: continuous; cat.: categorical variable; dyna: dynamic; val.: validate

2 Related works

Traffic forecasting models can be broadly divided into temporal modeling and spatiotemporal modeling. A wide array of temporal models has been applied to traffic prediction, progressing from classical methods, e.g., historical average (HA) and autoregressive integrated moving average (ARIMA) (Liu J and Guan, 2004; Yang et al., 2020), to deep learning architectures like long short-term memory (LSTM) and Transformers. These deep learning architectures have proven highly effective at capturing complex traffic dynamics (Hochreiter and Schmidhuber, 1997; Ma et al., 2015; Vaswani et al., 2017; Zhou et al., 2021). Spatiotemporal models incorporate spatial dependencies primarily through convolutional neural networks (CNNs) or graph convolutional networks (GCNs), which leads to foundational architectures like spatiotemporal GCNs (Yu et al., 2018; Song et al., 2020). Current research, however, focuses on more advanced techniques for simultaneous spatiotemporal modeling. Key innovations include learning dynamic graph structures with adaptive matrices, e.g., Graph WaveNet (Wu ZH et al., 2019), modeling continuous dynamics with ordinary differential equations, e.g., spatial-temporal graph ordinary differential equation neural network (STGODE) (Fang et al., 2021), and developing unified attention mechanisms that concurrently capture spatiotemporal correlations, e.g., T-Graphormer (Bai and Liu, 2025).

Although spatiotemporal models excel at learning from traffic data that are commonly sourced from sensor networks like PeMS (https://pems.dot.ca.gov), such as in Yu et al. (2018), Guo et al. (2019), and Song et al. (2020), their performance is inherently limited

by overlooking external factors. Consequently, although many studies now incorporate auxiliary data such as weather and time features (see more in the supplementary materials), their fusion methodologies exhibit critical bottlenecks. Specifically, current methods often rely on simplistic techniques like concatenation, failing to capture complex interdependencies. They also tend to treat auxiliary data as global variables, thereby ignoring crucial spatial heterogeneity (e.g., localized weather effects). Furthermore, these approaches typically lack the scalability and generalizability required to handle diverse or novel auxiliary data streams. These challenges underscore the fact that effective multi-source data utilization requires more than just data availability; advanced, scalable, and spatially aware fusion strategies are paramount. As shown in Table 1, although several valuable benchmark platforms exist, they are not tailored for multi-source traffic prediction. Platforms like time-series prediction platform (TSPP), time-series library (TSLib), and basic time series (BasicTS$^+$) focus on general time-series forecasting (Wu HX et al., 2023; Bączek et al., 2024; Shao et al., 2025). Conversely, even comprehensive traffic-specific libraries like LibCity (Wang et al., 2021) primarily handle core traffic data and timestamps, offering limited support for integrating diverse auxiliary data sources.

Consequently, a critical gap remains; there is no unified platform designed to systematically evaluate how different models leverage rich auxiliary data (e.g., weather and events) in traffic forecasting. This absence hinders the objective comparison of data fusion strategies, and slows progress in developing truly robust models. To address this, we develop MltAuxTSPP as a unified and extensible environment specifically for this purpose.

3 Multi-source auxiliary data fusion framework

3.1 Task formulation

In this work, the task of traffic state prediction with multi-source auxiliary data uses historical traffic states to forecast future traffic states, incorporating auxiliary data from both historical and future perspectives. Traffic state refers to the conditions of traffic flow on a road network, typically characterized by traffic volume, speed, density, and congestion levels. When no auxiliary data are used, the task reduces to single-source spatiotemporal forecasting. If spatial dependencies are ignored or flattened into features, it further simplifies to time-series forecasting.

Given the historical target variable $\boldsymbol{X} \in \mathbb{R}^{T\times n\times f}$, auxiliary variables $\boldsymbol{C}^{(1)}, \boldsymbol{C}^{(2)}, \ldots, \boldsymbol{C}^{(m)}$, and the road network $G = (N, E)$, the task is to predict a future sequence $\boldsymbol{X}^* \in \mathbb{R}^{h\times n\times f}$, where T is the historical input length, h is the length of the future horizon, n is the road network node count, f is the feature count, and m is the number of auxiliary variables. Each auxiliary variable $\boldsymbol{C}^{(j_c)} \in \mathbb{R}^{T^{(j_c)}\times n^{(j_c)}\times f^{(j_c)}}$ is characterized by its own temporal span $T^{(j_c)}$, node count $n^{(j_c)}$, and feature count $f^{(j_c)}$ ($j_c \in 1, 2, \cdots, m$). The graph G with node set N and edge set E is encoded as an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. Entries $A_{a,b}$ indicate either binary connectivity or weighted relationships (e.g., distance and travel time) between nodes N_a and N_b (a and $b \in [1, n]$). Notably, $\boldsymbol{A}$ is treated as one of the auxiliary variables.

Table 1 Comparison of benchmark platforms for time series and traffic prediction

Framework	Primary focus	Key feature/scope	Auxiliary data support
TSPP	General time series	Including nine models (e.g., temporal fusion Transformers (Lim et al., 2021)) and some traffic datasets	Limited
TSLib	General time series	Providing advanced models like Informer and TimesNet	Limited
BasicTS$^+$	General time series	Focusing on multi-variable, spatiotemporal, and long-term forecasting	Limited
LibCity	Traffic prediction	Comprehensive library with 60 models and 35 datasets for various traffic tasks	Limited
Ours	Multi-source traffic prediction	Unified environment with data containers and fusion modules for diverse auxiliary inputs	Yes

The task uses the model $\boldsymbol{M}: \{\boldsymbol{X}, \boldsymbol{C}^{(i)}\} \to \mathbb{R}^{t\times n\times f}$ $(i = 1, 2, \cdots, m)$ to predict the future sequence $\boldsymbol{X}^*$ and minimize the expected error $\mathbb{E}[\mathcal{L}(\boldsymbol{X}^*, \widehat{\boldsymbol{X}}^*)]$, where $\mathcal{L}$ measures the discrepancy between prediction $\boldsymbol{X}^*$ and the ground truth $\widehat{\boldsymbol{X}}^*$.

3.2 Data collection and preprocessing

Our study leverages two standard traffic benchmark datasets, METR-LA and PEMS-BAY (Li et al., 2018), augmented with external data to investigate their impact. The auxiliary data include three global weather datasets, ERA5-Land (https://cds.climate.copernicus.eu/datasets/reanalysis-era5-land?tab=overview), ERA5-HEAT (https://cds.climate.copernicus.eu/datasets/derived-utci-historical?tab=overview), and the global hourly-integrated surface database (GH-ISD) (https://www.ncei.noaa.gov/products/land-based-station/integrated-surface-database#tab-336), and three discrete temporal features, minute of the hour (MoH), hour of the day (HoD), and day of the week (DoW). To unify the spatiotemporal heterogeneity of these sources, we develop a preprocessing pipeline. First, all weather data are temporally interpolated to a 5-min frequency to match the traffic data. Second, all datasets are spatially converted to point-based representations, with weather data aligned to the nearest traffic sensor. Finally, we develop two enriched datasets in Table 2, and the detailed variable descriptions are provided in the supplementary materials.

3.3 Framework of the benchmarking system

We design the benchmarking system in Fig. 1 with the primary goal of integrating multi-source data and fairly comparing the performance of different models. The system adopts a top–down design approach that divides the entire architecture into several modules, each responsible for a specific function to achieve modularity. This design approach not only enhances the maintainability and scalability of the system, but also improves its flexibility and comprehensibility.

The system modules are categorized into two main types: core modules and auxiliary modules. The core modules include the data module, model module, trainer module, evaluator module, and executor module, whereas the auxiliary modules consist of the config and tuner modules. The core modules are implemented in Python, and the interfaces and behaviors of the modules are defined through abstract base classes. Through derived classes, these modules can adapt to a variety of scenario requirements. Specifically, the functions of each module are shown in Table 3.

Table 2 Summary of datasets

Source	Interval (min)	METR-LA		
		N_{step}	N_{node}	N_{var}
Traffic	5	34 272	207	2
ERA5-Land			10	10
ERA5-HEAT			4	2
GH-ISD			8	1
Source	**Interval (min)**	**PEMS-BAY**		
		N_{step}	N_{node}	N_{var}
Traffic	5	52 116	325	2
ERA5-Land			9	10
ERA5-HEAT			2	2
GH-ISD			5	1

N_{step}: number of steps; N_{node}: number of nodes; N_{var}: number of variables

The core and auxiliary modules are interconnected through the executor and Hydra (Meta Research, 2019). The executor instantiates the modules based on the configuration files, invokes their interfaces to complete the training and evaluation tasks, and then feeds the results back to Optuna (Akiba et al., 2019) for further optimization.

In this work, we address the fusion of multi-source auxiliary data using the pipeline shown in Fig. 2. First, we store the multi-source data in a UDC. Then, the data are transformed into a format suitable for model input through a fusion embedding module. The output of the module is a hidden representation that integrates the multi-source data, ensuring compatibility with the input requirements of the subsequent models. A more detailed explanation of this process will be provided in the following subsections.

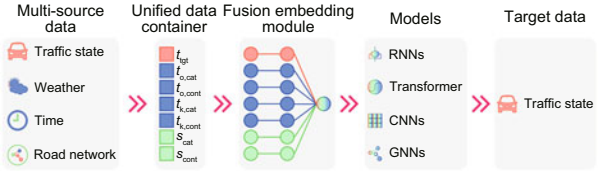

Fig. 2 Multi-source auxiliary data fusion framework for traffic state prediction. tgt: target; k: known; o: observed

Table 3 Description of the framework modules

Module	Description
Data	Responsible for preparing the datasets required by the models, including data integration, filtering, cleaning, and constructing UDCs
Model	Provides a series of reproducible or custom models to meet the requirements of prediction tasks
Trainer	Responsible for selecting optimization strategies and customizing relevant decisions to improve model performance
Evaluator	Assesses the accuracy and effectiveness of the model predictions
Executor	Controls the instantiation of modules based on the configuration and executes the training, inference, and evaluation processes using standardized interfaces
Config	Manages all framework-related parameters centrally through YAML files, providing structured and flexible control and adaptability
Tuner	Uses Optuna (Akiba et al., 2019) to select and apply the optimal configuration parameters after each module execution to optimize model performance

3.4 UDC

To manage and unify multi-source data, we use a UDC to store traffic state datasets with auxiliary information, which is an extension of the data aspects in TSPP (Bączek et al., 2024). The dataset can be categorized into the seven components in Table 4 based on the following attributes. Notably, the target variable is dynamic and unknown, and can be either continuous or categorical, whereas covariates may include multiple variables.

The data in our study are classified along four dimensions. By source, we distinguish between the target variables to be predicted and covariates. By temporal nature, data are either static (e.g., network topology) or dynamic. With respect to future availability, inputs can be known at prediction time (e.g., timestamps) or unknown (the target value and weather). Finally, by data type, variables are either continuous (real-valued) or categorical (discrete).

3.5 Fusion embedding modules

To improve the integration of multi-source data and facilitate model reuse, we propose an abstract fusion embedding module designed specifically for handling such data in Fig. 3. This module acts as an intermediary between the UDC and the model, with the primary objective of projecting multi-source data into a unified feature space for input into subsequent predictive models. We present three concrete instances of this abstract module: FlattenCat, temporal fusion Transformer (TFT), and TFT-gated weighted summation (TFT-GWS). FlattenCat standardizes variables through linear transformation, TFT (Lim et al., 2021) incorporates nonlinear processing and variable selection, and TFT-GWS is a simplified version of TFT that retains the core concepts while reducing computational overhead.

The abstract module consists of two main components: a variable formatting layer and a fusion layer. The variable formatting layer processes each variable through linear transformations, converting them into standardized variables for subsequent analysis. The fusion layer then integrates these standardized variables along the feature dimension,

Table 4 Components of UDC

Component	Source	Temporal	Observation	Data type
t_{tgt}	Target	Dynamic	Unknown	Continuous, categorical
s_{cont}	Covariates	Static	Known	Continuous
s_{cat}			Known	Categorical
$t_{\text{k,cont}}$		Dynamic	Known	Continuous
$t_{\text{k,cat}}$			Known	Categorical
$t_{\text{o,cont}}$			Unknown	Continuous
$t_{\text{o,cat}}$			Unknown	Categorical

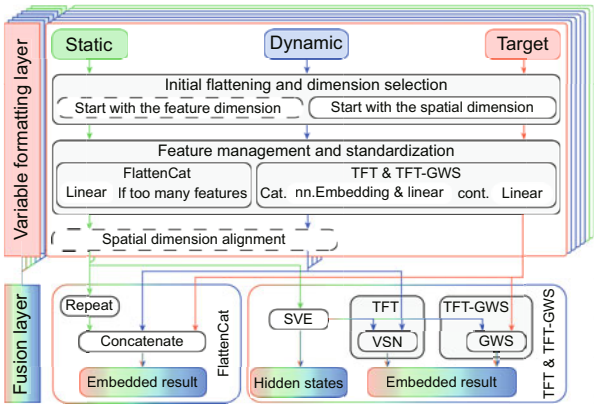

Fig. 3 Fusion embedding modules

producing a unified hidden representation that aligns with the model input specifications.

Consider the target variable $\boldsymbol{V}^{\mathrm{tgt}}$ and a set of p variables $V=\{\boldsymbol{V}^1,\boldsymbol{V}^2,\ldots,\boldsymbol{V}^p\}$. The target variable $\boldsymbol{V}^{\mathrm{tgt}}\in\mathbb{R}^{L^{\mathrm{tgt}}\times S^{\mathrm{tgt}}\times F^{\mathrm{tgt}}}$ has L^{tgt} time steps, S^{tgt} spatial nodes, and F^{tgt} features. Each variable $\boldsymbol{V}^i\in\mathbb{R}^{L^i\times S^i\times F^i\times\cdots}$ is multidimensional, where L^i is the temporal length, S^i is the number of spatial nodes, and F^i is the number of features. The ellipsis $(\ldots)$ denotes additional higher dimensions. For simplicity, we define the temporal window length of static variables as 1, dynamic variables in the historical dataset as T, and dynamic variables in the prediction window as h. Therefore, L^{tgt} and L^i can take values from $\{1,T,h\}$.

The fusion embedding module processes each auxiliary data source $\boldsymbol{V}^i$ through a three-stage pipeline to produce a standardized tensor $\boldsymbol{\xi}^i$ compatible with downstream models. The stages are as follows:

1. Initial flattening and dimension selection. First, each high-dimensional input tensor $\boldsymbol{V}^i\in\mathbb{R}^{L^i\times S^i\times F^i\times\cdots}$ is vectorized. This can be configured in two ways: (1) preserving the spatial dimension to yield a tensor $\boldsymbol{V'}^i\in\mathbb{R}^{L'^i\times S'^i\times F'^i}$, or (2) collapsing the spatial dimension to yield a flat feature vector $\boldsymbol{V'}^i\in\mathbb{R}^{F'^i}$.

2. Feature management and standardization. Next, the resulting feature dimension F'^i is projected to a target model dimension F_{m}. We provide two methods for this projection from $\boldsymbol{V'}^i_f$ to $\widetilde{\boldsymbol{V}}^i_f$, where f is the feature dimension for $\boldsymbol{V'}^i$: (1) FlattenCat conditionally applies a linear projection to reduce dimensionality as defined in Eq. (1); (2) TFT processes categorical and continuous features separately (using embeddings or linear layers) before projecting them to the target dimension F_{m}, as detailed in Eqs. (2) and (3):

$$\widetilde{\boldsymbol{V}}^i_f=\begin{cases}\mathrm{FC}(\boldsymbol{V'}^i_f)\in\mathbb{R}^{F_{\mathrm{m}}}, & \text{if } F'^i\geq F_{\mathrm{m}},\\ \boldsymbol{V'}^i_f\in\mathbb{R}^{F'^i}, & \text{if } F'^i<F_{\mathrm{m}},\end{cases}\tag{1}$$

$$\widetilde{\boldsymbol{V}}^i_f\in\mathbb{R}^{F_{\mathrm{m}}}=\mathrm{FC}(\mathrm{Flatten}(\mathrm{Embed}(\boldsymbol{V'}^i_{f,\mathrm{cat}}))),\tag{2}$$

$$\widetilde{\boldsymbol{V}}^i_f\in\mathbb{R}^{F_{\mathrm{m}}}=\mathrm{FC}(\boldsymbol{V'}^i_{f,\mathrm{cont}}),\tag{3}$$

where Flatten is matrix flattening, Embed is nn.Embedding, and FC is fully connected layers for DL.

3. Spatial dimension alignment (optional). Finally, if the spatial dimension S^i was preserved at the first stage, it is aligned to the target model's required spatial dimension S^{tgt} via a linear layer ($\mathrm{FC}:\mathbb{R}^{S^i}\to\mathbb{R}^{S^{\mathrm{tgt}}}$). This step is essential for downstream spatiotemporal models (e.g., T-GCN) that require spatially consistent inputs, but it is bypassed for purely temporal models (e.g., LSTM).

In the fusion layer, the three modules diverge in their methodologies. Consider the feature dimension of the formatted variable $\boldsymbol{\xi}^i_f\in\mathbb{R}^{F^i_f}$. FlattenCat extends the static values along the temporal axis and combines the formatted outputs from all variables to form a unified hidden representation $\boldsymbol{H}_f=[(\boldsymbol{\xi}^1_f)^{\mathrm{T}},(\boldsymbol{\xi}^2_f)^{\mathrm{T}},\ldots,(\boldsymbol{\xi}^p_f)^{\mathrm{T}}]^{\mathrm{T}}\in\mathbb{R}^{F_{\mathrm{H}}}$, where F_{H} is the sum of the individual feature dimensions, calculated as $F_{\mathrm{H}}=\sum_{i=1}^{p}F^i_f=\sum_{i=1}^{p}\min(F^i,F_{\mathrm{m}})$.

TFT processes formatted static variables using a static variable encoder (SVE), as shown in Fig. 4c. The SVE incorporates a variable selection network (VSN) and four gated residual networks (GRNs) to generate four static encodings: $\boldsymbol{c}_{\mathrm{s}}$, $\boldsymbol{c}_{\mathrm{e}}$, $\boldsymbol{c}_{\mathrm{h}}$, and $\boldsymbol{c}_{\mathrm{c}}$, as shown in Figs. 4a and 4d. Subsequently, TFT employs an additional VSN to process dynamic variables, including the target, and combines these

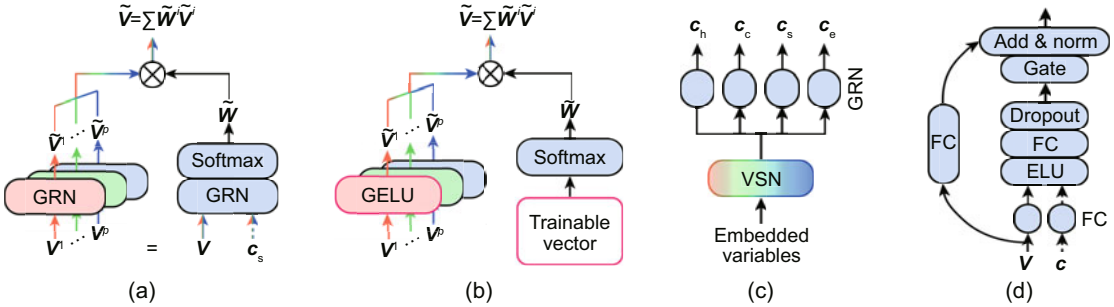

Fig. 4 The structures of VSN (a), GWS (b), SVE (c), and GRN (d). GWS: gated weighted summation; ELU: exponential linear unit

processed dynamic variables with the static encoding $\boldsymbol{c}_{\mathrm{s}}$.

The VSN implementation within TFT, crucial for identifying salient variables, follows three main steps, whose logical relationship is key to its function:

1. Individual feature transformation. Each formatted variable $\boldsymbol{\xi}_f^i$ is individually processed by a GRN, which applies a gating mechanism to filter out irrelevant variables for the prediction task. That is, each GRN generates a transformed variable $\widetilde{\boldsymbol{\xi}}_f^i \in \mathbb{R}^{F_{\mathrm{m}}}$, as shown in Eq. (4):

$$\widetilde{\boldsymbol{\xi}}_f^i = \mathrm{GRN}(\boldsymbol{\xi}_f^i, \boldsymbol{c}_{\mathrm{s}}) \in \mathbb{R}^{F_{\mathrm{m}}}. \tag{4}$$

2. Variable weight generation. To determine the relevance of each variable, the VSN computes dynamic attention weights. As described in Eq. (5), all original formatted input variables $\{\boldsymbol{\xi}_f^1, \boldsymbol{\xi}_f^2, \ldots, \boldsymbol{\xi}_f^p\}$ are first stacked and flattened into a single vector $\boldsymbol{\Xi}_f \in \mathbb{R}^{p \times F_{\mathrm{m}}}$. This vector is then passed through another GRN with a context vector $\boldsymbol{c}_{\mathrm{s}}$ if providing static variables, and the output is fed into a Softmax function. This yields a set of dynamic weights $\widetilde{\boldsymbol{w}}^1, \widetilde{\boldsymbol{w}}^2, \ldots, \widetilde{\boldsymbol{w}}^p$, where each $\widetilde{\boldsymbol{w}}^i \in \mathbb{R}^p$ corresponds to the importance of the i^{th} variable. These weights are data-dependent, meaning that they are recalculated for each input instance.

$$\begin{aligned} &\boldsymbol{\Xi}_f = \mathrm{Flatten}([\boldsymbol{\xi}_f^1, \boldsymbol{\xi}_f^2, \ldots, \boldsymbol{\xi}_f^p]) \in \mathbb{R}^{p \times F_{\mathrm{m}}}, \\ &\widetilde{\boldsymbol{W}} = \mathrm{Softmax}(\mathrm{GRN}(\widetilde{\boldsymbol{\Xi}}_f)) \in \mathbb{R}^p. \end{aligned} \tag{5}$$

3. Weighted summation. Finally, the VSN produces its output $\boldsymbol{H}_f$ by performing a weighted summation of the transformed features $\widetilde{\boldsymbol{\xi}}_f^i$ (from Eq. (4)), using the generated variable weights $\widetilde{\boldsymbol{w}}^i$ (from Eq. (5)). This process, illustrated in Eq. (6), effectively selects and combines the most relevant features:

$$\boldsymbol{H}_f = \sum_{i=1}^{p} \widetilde{\boldsymbol{w}}^i \widetilde{\boldsymbol{\xi}}_f^i. \tag{6}$$

In practice, we observe that TFT's VSN, particularly the GRNs within it, significantly contributes to memory usage, especially with a large number of variables. To address this inefficiency, we introduce the GWS module, as depicted in Fig. 4b. The GWS module modifies the VSN architecture for improved efficiency. The key distinctions between the GWS module and TFT's VSN are as follows:

1. Modification to individual feature transformation in Eq. (4). The computationally intensive GRN is replaced with a simple gated unit, i.e., Gaussian error linear unit (GELU) activation function. The transformation becomes $\widetilde{\boldsymbol{\xi}}_f^i = \mathrm{GELU}(\boldsymbol{\xi}_f^i)$. This significantly reduces parameters and computational load compared to a full GRN.

2. Modification to variable weight generation in Eq. (5). Weight generation is simplified by making weights data-independent. It uses a learnable (trainable) parameter vector $\boldsymbol{W} \in \mathbb{R}^p$. The variable selection weights $\widetilde{\boldsymbol{W}}$ are then obtained by applying Softmax directly to this vector: $\widetilde{\boldsymbol{W}} = \mathrm{Softmax}(\boldsymbol{W})$. These weights are learned during the training process, and remain fixed thereafter for inference, rather than being re-computed for each input.

3. No change in weighted summation in Eq. (6).

3.6 Models with fusion embedding results

We investigate the integration of embedding modules in canonical time series and spatiotemporal prediction models, focusing on three key models. We select sequence to sequence (Seq2Seq) (Sutskever et al., 2014) as a representative of RNNs and Informer (Zhou et al., 2021) as an example of Transformer-based approaches. Given the graph-structured nature of our data, we choose temporal graph convolutional network (T-GCN) (Zhao et al., 2020) as the main spatiotemporal model and adapt its architecture like Seq2Seq for multi-step forecasting, as shown in Fig. 5c. We also choose Graph WaveNet (Wu ZH et al., 2019).

In our model design, we observe that the original Seq2Seq and Informer models intentionally disregard or occlude variables within the future window. This design choice stems from their primary use in forecasting a single unknown variable, for which the true value in the future window is unavailable. However, in traffic-state prediction tasks that involve multiple auxiliary variables, covariates from various data sources within the future window are accessible and can potentially improve predictive accuracy. To address this, we refine the model to better use information from the future window as shown in Fig. 5.

Specifically, we use both the hidden state and the output from the embedding module as the input to the decoder of Seq2Seq instead of only the hidden state. For Informer, we remove the mask in the decoder's first layer, allowing it to receive information from the future window. Additionally, when extracting samples from the dataset for the future window,

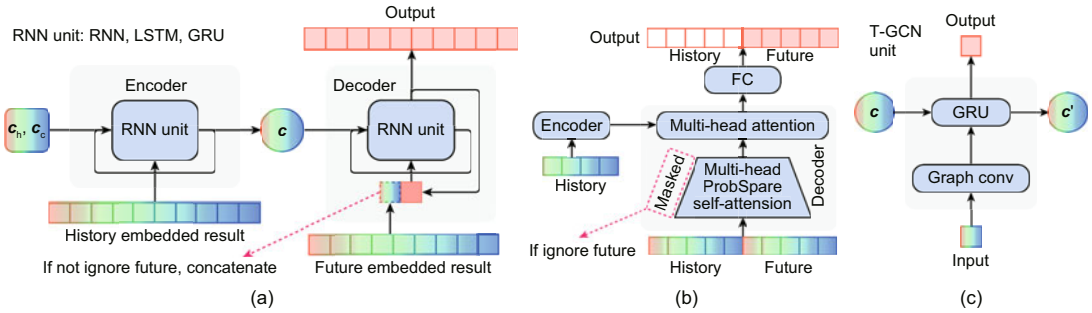

Fig. 5 Fusion embedding modules and models: (a) Seq2Seq; (b) Informer; (c) T-GCN unit. Conv: convolution

we assign a value of 0 to unknown variables to simulate the uncertainty encountered in the real-world scenarios.

4 Experiments

4.1 Experimental setup

We conduct experiments on the augmented PEMS-BAY and METR-LA datasets detailed in Section 3.2. All input variables are normalized using Z-score, and each dataset is divided into training, validation, and test sets using a 7:2:1 ratio. Model performance is evaluated using mean absolute error (MAE) and mean squared error (MSE).

All experiments are conducted on a server equipped with an NVIDIA RTX 4090 GPU (24 GB) and an Intel Xeon Gold 6430 CPU, running Ubuntu 22.04. Our framework is built with Python 3.12.3 and PyTorch 2.3.0 (CUDA 12.1). To enhance training efficiency, we use automatic mixed precision (AMP) via the torch.amp package.

Our experimental process consists of two main stages: parameter tuning and final evaluation. To ensure a fair comparison, we first tune hyperparameters for baseline models (Seq2Seq, Informer, and T-GCN). At this stage, models are trained for up to three epochs using only traffic variables as input, consistent with their original designs. We use the Adam optimizer with a learning rate of 0.001 and a batch size of 512, and employ early stopping to prevent overfitting. For the main evaluation, we assess model performance under various conditions summarized in Table 5. To ensure robustness, each experiment is performed 10 times with different random seeds. Unless otherwise specified, all models are trained for up to 15 epochs using the Adam optimizer (with learning rate 0.001 and batch size 512) with early stopping. Minor adjustments to batch size and AMP usage are made on a case-by-case basis to prevent out-of-memory errors.

4.2 Performance and resource utilization of fusion embedding modules

In Section 3.5, we introduce three fusion embedding modules: FlattenCat, TFT, and TFT-GWS. To evaluate their performance and resource utilization, we measure the time and memory costs of each module during both inference and training, as detailed in Table 6. Our analysis focuses on varying sequence lengths, spatial dimensions, and the number of external features to provide a comprehensive understanding of their performance characteristics. We do not reiterate the computational overhead of subsequent models (e.g., Informer and T-GCN) because this has been extensively covered in previous research (Shao et al., 2025). When running the experiments, each module receives multiple time-series variables as the input, spanning historical and future windows of length L_{w}. Specifically, each variable has a shape of $(16, 2L_{\mathrm{w}}, N_{\mathrm{s}}, 1)$, encompassing one target component and N_{Co} variables for each covariate component as detailed in Table 4. When the retaining spatial dimension (RSD) is enabled, the module outputs a variable of shape $(16, 2L_{\mathrm{w}}, N_{\mathrm{s}}, 512)$; otherwise, the output shape is $(16, 2L_{\mathrm{w}}, 512)$.

4.3 Basic performance of forecasting

The baseline model performance on the PEMS-BAY and METR-LR datasets using only traffic data is presented in Table 7 (more in the supplementary materials). The analysis highlights four primary findings: (1) The prediction error increases as

Table 5 Experimental conditions for model evaluation

Condition	Setting
Number of nodes (N_s)	1-1, 4-1, 16-x, all (n-i denotes n nodes with index intervals i, and $x \in \{1, 2, 4, 8\}$)
Time window (L_w)	$L_w \in \{12, 24, 48, 96\}$ (input and output windows are identical)
Auxiliary variables (Co)	Traffic-only (Co = No); All auxiliary variables (Co = All)
Embedding modules	FlattenCat, TFT, TFT-GWS
Model perception of future window	With or without future window information
Model after fusion embedding	Seq2Seq, Informer, T-GCN, graph WaveNet

the length of the forecast horizon increases; (2) The model performance is highly sensitive to the number of nodes used, with different architectures responding differently to the network scale; (3) The choice of embedding module is critical because nonlinear embeddings provide a significant advantage for long-term forecasting over simpler methods; (4) As the number of nodes increases, the complex spatial dependencies can act as noise, and the resulting modeling challenges can overwhelm the informational benefits, ultimately harming predictive accuracy.

4.4 Improvement with covariates

We evaluate the impact of covariates by comparing prediction performance with and without them under identical experimental configurations (Section 4.3). As shown in Table 8, covariates significantly improve the accuracy for long-term forecasts and in low-complexity, single-node settings, highlighting their value for extended prediction horizons.

The impact of covariates varies with prediction horizon and node complexity. For short horizons (e.g., 12 and 24 steps), covariates offer minimal or even detrimental effects. This is likely due to the sampling frequency mismatch between high-frequency traffic data and low-frequency covariates like weather data. Conversely, their benefits are more pronounced for longer horizons (e.g., 48 and 96 steps), where their inclusion consistently enhances the model performance. The positive impact of covariates is most significant in single-node tasks. In multi-node scenarios, however, models can still effectively leverage covariate information by capturing interactions across different nodes, leading to performance gains.

The FlattenCat method is less stable and effective than TFT and TFT-GWS, especially in short-term and full-node scenarios. FlattenCat exhibits inconsistent performance; although effective in some long-term, low-node-count tasks, it shows marked instability in short-term and full-node scenarios. Specifically, it causes significant performance degradation in short-term predictions (12 and 24 steps), and leads to training anomalies in full-node configurations. In contrast, TFT and TFT-GWS consistently outperform FlattenCat and remain stable across most scenarios. Their advantage is particularly notable in single-node configurations. Although their performance gains are less pronounced in full-node tasks, they avoid the anomalous behavior seen with FlattenCat. It is worth noting that TFT-GWS occasionally underperforms TFT, particularly in short-term predictions, which we attribute to the potential inefficiencies in its gating mechanism.

4.5 Impact of covariates in the future window

We conduct a comparative analysis of prediction tasks that consider future window information with those that do not. The analysis in Table 9 shows that incorporating future window information, particularly with known covariates, significantly improves prediction performance.

TFT and TFT-GWS benefit from future covariates, whereas FlattenCat struggles, often showing adverse effects due to weak covariate-target correlations. When using the TFT and TFT-GWS embedding modules, known covariates in the future window generally improve prediction accuracy, enabling the model to more effectively capture the evolving patterns of the target data. However, for the FlattenCat module, the inclusion of known covariates in the future window exhibits minimal impact on the prediction performance, and in some instances, it even adversely affects the performance, especially in the Informer model. This can be attributed to covariates with low correlation to the target data, which tend to interfere more with the model than provide valuable information. Future window information offers minimal benefits without covariates, and may introduce biases that hinder prediction performance.

Table 6 Performance and resource utilization of fusion embedding modules

Embedding module	N_s	RSD	Number of parameters					
			N_{Co}=0		1		4	
			L_w=6	48	6	48	6	48
FlattenCat	8	False	0	0	0	0	0	0
		True	0	0	432	432	1.7×10^3	1.7×10^3
	16	False	0	0	0	0	0	0
		True	0	0	1.6×10^3	1.6×10^3	6.5×10^3	6.5×10^3
TFT	8	False	2.1×10^6	2.1×10^6	17.3×10^6	17.3×10^6	50.7×10^6	50.7×10^6
		True	2.1×10^6	2.1×10^6	16.9×10^6	16.9×10^6	49.3×10^6	49.3×10^6
	16	False	2.1×10^6	2.1×10^6	17.7×10^6	17.7×10^6	52.3×10^6	52.3×10^6
		True	2.1×10^6	2.1×10^6	16.9×10^6	16.9×10^6	49.3×10^6	49.3×10^6
TFT-GWS	8	False	4.6×10^3	4.6×10^3	6.7×10^6	6.7×10^6	14.3×10^6	14.3×10^6
		True	1.0×10^3	1.0×10^3	6.4×10^6	6.4×10^6	12.9×10^6	12.9×10^6
	16	False	8.7×10^3	8.7×10^3	7.1×10^6	7.1×10^6	16.0×10^6	16.0×10^6
		True	1.0×10^3	1.0×10^3	6.4×10^6	6.4×10^6	12.9×10^6	12.9×10^6

Embedding module	N_s	RSD	Inference											
			Time (ms)						Memory (MB)					
			N_{Co}=0		1		4		N_{Co}=0		1		4	
			L_w=6	48	6	48	6	48	6	48	6	48	6	48
FlattenCat	8	False	0.1	0.1	0.4	0.4	0.7	0.7	0.0	0.2	0.2	1.3	0.6	4.6
		True	0.1	0.1	0.6	0.6	1.6	1.5	0.0	0.2	1.1	1.9	1.4	4.6
	16	False	0.1	0.1	0.4	0.4	0.7	0.7	0.1	0.5	0.3	2.6	1.1	9.9
		True	0.1	0.1	0.7	0.6	1.5	1.4	0.1	0.5	1.2	2.9	1.8	9.9
TFT	8	False	1.2	1.2	4.4	5.3	13.3	10.4	2.7	16.6	8.9	62.4	26.6	208.8
		True	1.2	2.0	4.8	11.6	14.6	40.9	16.5	132.1	63.3	482.1	209.5	1637.0
	16	False	1.0	1.0	5.4	5.4	13.4	11.9	2.7	16.7	9.0	62.9	26.8	210.4
		True	1.1	3.7	6.1	21.8	14.3	81.3	33.0	264.3	126.9	964.2	418.2	3274.0
TFT-GWS	8	False	0.3	0.4	2.2	2.8	4.8	4.7	1.8	12.1	9.2	67.9	29.9	232.7
		True	0.4	0.6	3.3	5.6	5.9	22.1	12.0	96.1	69.3	530.0	233.4	1828.6
	16	False	0.4	0.4	2.8	2.8	4.7	5.6	1.8	12.2	9.3	68.4	30.1	234.3
		True	0.4	1.4	4.0	11.3	7.4	46.9	24.0	192.2	138.9	1059.9	466.1	3657.2

Embedding module	N_s	RSD	Training											
			Time (ms)						Memory (MB)					
			N_{Co}=0		1		4		N_{Co}=0		1		4	
			L_w=6	48	6	48	6	48	6	48	6	48	6	48
FlattenCat	8	False												
		True			2.5	2.7	5.5	5.7			1.1	2.7	1.5	10.2
	16	False												
		True			2.6	2.4	6.0	5.3			1.3	5.4	2.5	20.9
TFT	8	False	5.7	6.4	21.3	20.0	53.7	48.3	9.2	32.8	55.9	121.7	183.4	409.1
		True	6.1	6.4	20.9	35.4	60.2	146.0	32.3	324.4	151.8	1462.6	510.5	5049.4
	16	False	6.6	5.5	20.0	23.1	54.8	51.4	9.2	32.9	57.5	126.1	189.8	428.6
		True	5.7	11.6	24.2	67.4	56.2	283.3	76.6	654.9	351.4	2976.4	1179.2	1 0273.7
TFT-GWS	8	False	1.9	2.1	6.2	7.8	15.3	17.8	3.4	27.1	11.3	87.3	34.3	293.8
		True	1.8	2.1	7.8	17.8	20.4	97.5	27.0	216.1	117.4	840.8	396.2	2961.1
	16	False	1.9	2.1	7.2	7.5	16.9	15.8	3.4	27.2	10.5	91.2	31.3	313.3
		True	2.2	4.4	9.9	35.6	28.8	197.3	54.0	432.2	235.2	1680.3	788.5	5921.0

N_{Co}: number of variables per covariate component in Table 4 (0 for no covariate, 1 for 6 covariates, and 4 for 24 covariates). RSD: retaining spatial dimension (false if not retained, true if retained)

Table 7 Forecast performance without covariates or future information in the PEMS-BAY dataset

Model	Embedding module	N_s	Forecast performance							
			L_w=12		24		48		96	
			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Seq2Seq	FlattenCat	1-1	0.2967	0.2702	0.3750	0.3148	0.5221	0.3949	0.6479	0.4398
		16-1	0.2864	0.2681	0.4540	0.3360	0.5574	0.3812	0.5541	0.3880
		16-4	0.1875	0.2050	0.2624	0.2421	0.3250	0.2810	0.3782	0.3083
		All	**0.2841**	**0.2762**	**0.3178**	**0.2905**	**0.3810**	**0.3281**	0.3867	0.3271
	TFT	1-1	0.3065	0.2673	0.3663	0.3105	0.5422	0.4024	0.6146	0.4245
		16-1	0.3067	0.2766	0.4266	0.3234	0.5638	0.3879	0.5541	0.3851
		16-4	0.1964	0.2065	0.2645	0.2430	0.3378	0.2882	0.3873	0.3132
		All	0.2890	0.2780	0.3194	0.2917	0.3817	0.3296	**0.3803**	**0.3224**
	TFT-GWS	1-1	0.2910	0.2629	0.3645	0.3078	**0.5069**	0.3850	**0.5767**	**0.3964**
		16-1	0.3096	0.2772	0.4507	0.3295	0.5516	0.3833	**0.5266**	**0.3687**
		16-4	0.1974	0.2094	0.2602	0.2418	**0.3217**	**0.2778**	**0.3698**	**0.3055**
		All	0.2916	0.2767	0.3216	0.2912	0.3834	0.3302	0.3857	0.3247
Informer	FlattenCat	1-1	0.3140	0.2770	0.4252	0.3352	0.5931	0.4249	0.6496	0.4424
		16-1	0.3215	0.2936	0.4859	0.3569	0.5775	0.3990	0.5835	0.4030
		16-4	0.2136	0.2234	0.2885	0.2605	0.3921	0.3184	0.4073	0.3359
		All	0.3061	0.2908	0.3481	0.3118	0.3901	0.3342	0.3847	0.3300
	TFT	1-1	0.3267	0.2879	0.4361	0.3421	0.5885	0.4160	0.6416	0.4393
		16-1	0.3509	0.3066	0.4943	0.3543	0.5993	0.4080	0.5940	0.4074
		16-4	0.2205	0.2275	0.2880	0.2587	0.4117	0.3247	0.4179	0.3412
		All	0.3106	0.2914	0.3386	0.3043	0.4220	0.3494	0.3841	0.3318
	TFT-GWS	1-1	0.3120	0.2775	0.4245	0.3369	0.5708	0.4161	0.6294	0.4319
		16-1	0.4076	0.3147	0.5071	0.3612	0.5663	0.3951	0.5714	0.3950
		16-4	0.2242	0.2321	0.2902	0.2618	0.3488	0.2950	0.4020	0.3283
		All	0.3175	0.2964	0.3464	0.3120	0.3912	0.3357	0.3814	0.3291
T-GCN	FlattenCat	1-1	0.3035	0.2777	0.3730	0.2959	0.5411	0.3951	0.7115	0.4678
		16-1	0.2880	0.2677	0.4683	0.3677	0.7011	0.4772	0.9316	0.5682
		16-4	0.2454	0.2265	0.4009	0.3071	0.6237	0.4003	0.8116	0.4885
	TFT	1-1	0.2892	0.2667	**0.3531**	**0.2927**	0.5205	0.3748	0.6151	0.3995
		16-1	0.2614	0.2534	0.4213	0.3456	0.6394	0.4411	0.8651	0.5424
		16-4	0.2242	0.2153	0.3771	0.2894	0.5850	0.3710	0.7847	0.4747
	TFT-GWS	1-1	**0.2839**	**0.2586**	0.3554	0.2946	0.5209	**0.3731**	0.6579	0.4447
		16-1	0.2636	0.2512	0.4308	0.3482	0.6521	0.4644	0.8797	0.5495
		16-4	0.2313	0.2159	0.3867	0.2943	0.5994	0.3922	0.7958	0.4853
Graph WaveNet	FlattenCat	1-1	0.3002	0.2633	0.4449	0.3372	0.5529	0.4001	0.6642	0.4382
		16-1	**0.2299**	**0.2283**	**0.3449**	**0.2892**	**0.4979**	**0.3468**	0.5751	0.4011
		16-4	**0.1794**	**0.1897**	**0.2383**	0.2242	0.3592	0.2845	0.4367	0.3344
	TFT	1-1	0.3278	0.2889	0.4142	0.3178	0.5237	0.3938	0.6281	0.4163
		16-1	0.2386	0.2331	0.3579	0.2897	0.5262	0.3614	0.5566	0.3867
		16-4	0.1933	0.1943	0.2422	**0.2222**	0.3740	0.2920	0.4395	0.3277
	TFT-GWS	1-1	0.3274	0.3075	0.4355	0.3341	0.5874	0.4215	0.6572	0.4244
		16-1	0.2493	0.2564	0.3869	0.3133	0.5337	0.3673	0.5710	0.3948
		16-4	0.1998	0.2134	0.2604	0.2391	0.3676	0.2919	0.4366	0.3292

The best results within the same window and node configuration are in bold

Table 8 Relative forecast performance gain with covariates in the PEMS-BAY dataset

Model	Embedding module	N_s	Forecast performance gain							
			L_w=12		24		48		96	
			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Seq2Seq	FlattenCat	1-1	**2.28**	−5.56	**0.08**	−3.05	**4.96**	**0.71**	**15.99**	−0.55
		16-1	−46.26	−16.85	−7.69	−3.56	**2.56**	**0.72**	**5.88**	**4.98**
		All	−275.81	−110.98	−237.09	−100.99	−183.22	−78.76	−176.47	−78.77
	TFT	1-1	**13.21**	**5.88**	**5.79**	**3.34**	**31.39**	**17.95**	**43.00**	**22.74**
		16-1	**0.19**	−0.00	−3.23	**0.09**	**13.05**	**10.14**	**13.26**	**11.88**
		All	**0.42**	**0.66**	**3.15**	**1.95**	**10.96**	**6.96**	**10.13**	**5.29**
	TFT-GWS	1-1	**5.59**	**0.46**	**5.41**	−1.08	**24.07**	**9.69**	**36.15**	**13.33**
		16-1	−41.74	−14.29	−6.95	−2.22	**9.05**	**8.04**	**7.49**	**5.98**
		All	−1.80	−1.88	**0.61**	−0.63	**9.75**	**6.63**	**10.18**	**3.89**
Informer	FlattenCat	1-1	−3.83	−10.12	−0.50	−4.46	**14.35**	**6.69**	**22.63**	**3.90**
		16-1	−38.03	−11.90	−2.84	−0.91	**7.38**	**5.04**	**7.43**	**6.44**
		All	−40.14	−23.43	−50.60	−29.83	−139.86	−61.33	−46.46	−24.30
	TFT	1-1	**10.85**	**4.36**	**10.34**	**2.25**	**32.53**	**16.46**	**37.18**	**18.11**
		16-1	−10.36	−1.20	**6.61**	**4.74**	**19.64**	**15.00**	**16.31**	**12.79**
		All	**4.56**	**2.50**	**3.90**	**1.64**	**19.23**	**11.53**	**8.26**	**4.43**
	TFT-GWS	1-1	**5.68**	−1.34	**17.03**	**4.96**	**35.30**	**20.55**	**35.88**	**15.16**
		16-1	−10.08	−3.52	**7.18**	**7.18**	**16.55**	**14.01**	**14.10**	**11.91**
		All	**3.35**	**2.00**	**3.34**	**2.17**	**9.08**	**5.95**	**4.01**	**1.61**
T-GCN	FlattenCat	1-1	**7.69**	**1.15**	**7.95**	−4.44	**26.67**	**13.73**	**39.14**	**22.71**
		16-1	**5.79**	−3.43	**8.48**	**0.57**	**17.87**	**10.78**	**23.99**	**16.52**
	TFT	1-1	**18.83**	**9.13**	**10.08**	**1.13**	**30.54**	**14.73**	**35.22**	**14.92**
		16-1	**11.60**	**4.64**	**21.70**	**13.58**	**30.42**	**19.21**	**40.49**	**30.26**
	TFT-GWS	1-1	−1.76	−9.90	**3.40**	−8.58	**23.22**	**7.08**	**39.22**	**23.18**
		16-1	**0.63**	−8.72	**5.89**	−0.07	**14.80**	**9.95**	**19.05**	**12.80**
Graph WaveNet	FlattenCat	1-1	**5.46**	−3.11	**13.89**	**2.16**	**12.18**	**8.38**	**15.27**	**5.07**
		16-1	−0.39	−2.02	**1.09**	**1.73**	**12.60**	**6.67**	**15.59**	**11.83**
	TFT	1-1	**16.16**	**4.43**	−14.48	−16.44	−6.30	−3.58	−27.52	−19.43
		16-1	**2.12**	**0.07**	**6.36**	**0.58**	**12.75**	**5.66**	**2.13**	**1.67**
	TFT-GWS	1-1	**0.20**	−2.42	**11.75**	**4.75**	**7.58**	**10.61**	−40.02	−18.22
		16-1	−19.56	−7.75	−3.10	−1.76	**14.42**	**5.92**	**3.77**	**3.17**

The percentage improvement is calculated as $v = -(m_{w/} - m_{w/o})/m_{w/o} \times 100\%$, where $m_{w/}$ and $m_{w/o}$ are metrics with and without future covariates, respectively. Positive improvements are in bold

Moreover, we observe that the benefits of future window information are marginal in the absence of covariates, and it even results in a decline in the prediction performance in some cases. This can be attributed to biases inherent in the embedding module, which interfere with the ability of the model to learn the target data.

4.6 Variable importance

To understand variable importance, we first note that simple statistical measures, such as Pearson correlation and maximum information coefficient (MIC) (Reshef et al., 2011; Albanese et al., 2013), reveal weak relationships between traffic speed and most auxiliary variables (Fig. 6), except for the MIC between traffic speed and HoD or DoW. However, analyzing the learned variable selection weights from our fusion modules reveals a more nuanced and interpretable picture, as shown in Fig. 7.

The model correctly assigns the highest importance to the historical target variable. More revealingly, it identifies the key predictive features that

Table 9 Relative forecast performance gain with future information in the PEMS-BAY dataset

Co	Model	Embedding module	Forecast performance gain							
			L_{w}=12		24		48		96	
			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
All	Seq2Seq	FlattenCat	**2.22**	**1.21**	**2.79**	**1.49**	**3.41**	**1.81**	**9.16**	**6.74**
		TFT	**9.19**	**5.52**	**12.85**	**8.18**	**14.27**	**11.23**	**10.29**	**9.50**
		TFT-GWS	**4.87**	**3.09**	**10.46**	**6.61**	**12.93**	**10.17**	**13.41**	**11.30**
	Informer	FlattenCat	−0.34	−0.17	**3.57**	**1.75**	**4.50**	**2.21**	−11.24	−4.55
		TFT	**7.95**	**3.84**	**4.85**	**3.74**	**6.42**	**5.43**	**12.00**	**10.14**
		TFT-GWS	**0.46**	**0.21**	**1.50**	**1.82**	**2.53**	**3.17**	**10.40**	**8.56**
	T-GCN	FlattenCat	**4.78**	**3.51**	**5.26**	**3.64**	**8.93**	**6.46**	**19.95**	**12.49**
		TFT	**10.72**	**6.73**	**15.04**	**10.73**	**20.22**	**15.94**	**21.89**	**16.90**
		TFT-GWS	**8.76**	**5.79**	**15.02**	**10.58**	**21.64**	**16.38**	**31.37**	**22.21**
	Graph WaveNet	FlattenCat	**5.26**	**0.07**	**3.19**	−0.70	**0.26**	−3.09	−9.11	−3.62
		TFT	−3.44	−1.91	**4.91**	−0.36	**7.41**	**1.71**	**15.75**	**10.76**
		TFT-GWS	**7.14**	**4.26**	**1.56**	−1.10	**5.67**	**1.02**	**11.21**	**6.31**
No	Seq2Seq	FlattenCat	−0.61	−0.18	−0.10	−0.09	**0.78**	**0.18**	**1.32**	**1.22**
		TFT	−1.28	−1.87	−0.98	−1.00	**3.35**	**2.20**	**1.58**	**2.01**
		TFT-GWS	−6.10	−3.16	−2.23	−1.58	**0.44**	**0.30**	−2.64	−1.46
	Informer	FlattenCat	**1.19**	**0.65**	**0.17**	−0.24	−0.54	−0.62	−0.64	−0.53
		TFT	**3.38**	**1.43**	−2.02	−1.76	−5.73	−4.23	−5.53	−4.37
		TFT-GWS	**4.80**	**1.99**	**0.52**	**0.03**	−2.16	−1.29	−1.00	−1.32
	T-GCN	FlattenCat	**0.02**	**0.07**	−0.71	−0.79	−0.07	**0.12**	**0.89**	**0.14**
		TFT	−0.69	−0.39	**1.43**	**1.01**	**0.82**	−0.57	**4.60**	**1.07**
		TFT-GWS	**0.02**	**0.86**	**0.03**	**0.66**	−0.53	−0.76	**4.99**	**3.03**
	Graph WaveNet	FlattenCat	−4.25	−5.32	−1.61	−1.86	**0.22**	−0.07	**0.53**	**0.71**
		TFT	**2.11**	−0.93	**1.40**	−3.88	−2.56	−1.57	−0.64	−0.74
		TFT-GWS	−0.84	**2.83**	**3.00**	**0.44**	**1.54**	**0.04**	−2.64	−3.20

The percentage improvement is calculated as $v = -(m_{\mathrm{w}/} - m_{\mathrm{w/o}})/m_{\mathrm{w/o}} \times 100\%$, where $m_{\mathrm{w}/}$ and $m_{\mathrm{w/o}}$ are metrics with and without future information, respectively. Positive improvements are in bold

simple correlations miss. For instance, it gives high weight to future HoD and DoW and to historical precipitation (tp) and visibility (vd). The importance of these variables is empirically supported: Traffic patterns are strongly tied to DoW (Fig. 8), and adverse weather is known to impact driving speed (FHWA, 2024). This demonstrates that the model learns meaningful, nonlinear relationships beyond simple statistical correlation. Furthermore, these weights allow for a comparative analysis of the embedding modules. The weights in the TFT-GWS module exhibit a smaller dynamic range than those in the TFT module. This suggests that TFT is more selective in focusing on critical variables, whereas TFT-GWS gives more even consideration to all inputs, which could make it more susceptible to noise from irrelevant features and negatively impact performance.

5 Conclusions

In this study, we introduce MltAuxTSPP, an innovative benchmarking framework designed to enhance traffic state prediction by integrating multi-source auxiliary data. Through a UDC and adaptable fusion embedding modules, MltAuxTSPP effectively addresses key challenges in data utilization and scalability. Our experimental results demonstrate that incorporating auxiliary data, particularly weather and temporal features, significantly improves long-term forecast accuracy. This work validates the efficacy of fusion embedding modules (e.g., FlattenCat, TFT, and TFT-GWS) in transforming heterogeneous data into a unified, predictive feature space. The comparative analysis of these methods offers valuable insights for model selection. Ultimately, MltAuxTSPP provides a robust and extensible framework that not only advances prediction accuracy but also serves as a critical tool for developing more resilient and efficient strategies for ITS. Although this study shows promising results, it has several limitations. First, computational overhead increases significantly with large-scale, multi-source data. Second, the model's sensitivity to the quality of auxiliary data poses a challenge. Although

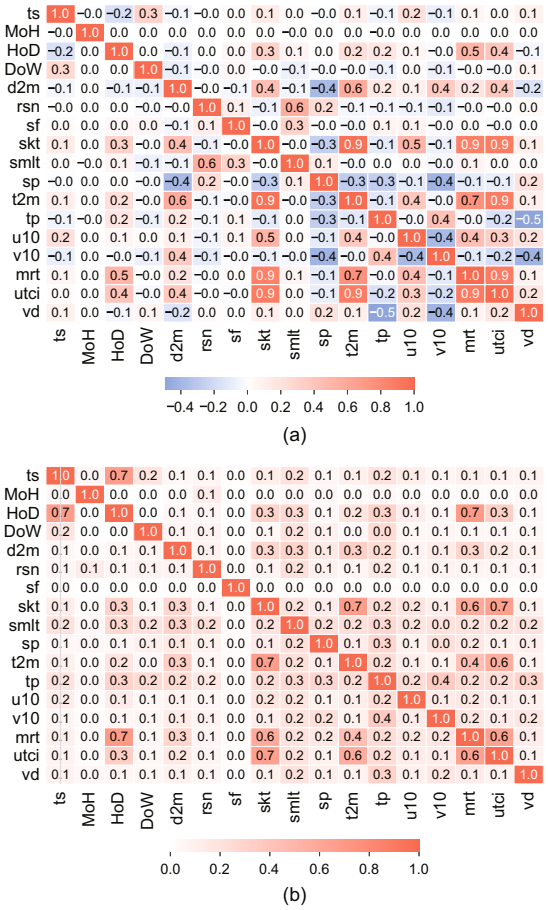

Fig. 6 Correlation between traffic state and covariates: (a) Pearson; (b) MIC. ts: traffic speed; d2m: 2-m dewpoint temperature; rsn: mass of snow per cubic meter in the snow layer; sf: accumulated total snow that has fallen to the Earth's surface; skt: skin temperature; smlt: melting of snow averaged over the grid box; sp: surface pressure; t2m: 2-m temperature; tp: total precipitation; u10: 10-m u-component of wind; v10: 10-m v-component of wind; mrt: mean radiant temperature; utci: universal thermal climate index; vd: visibility distance

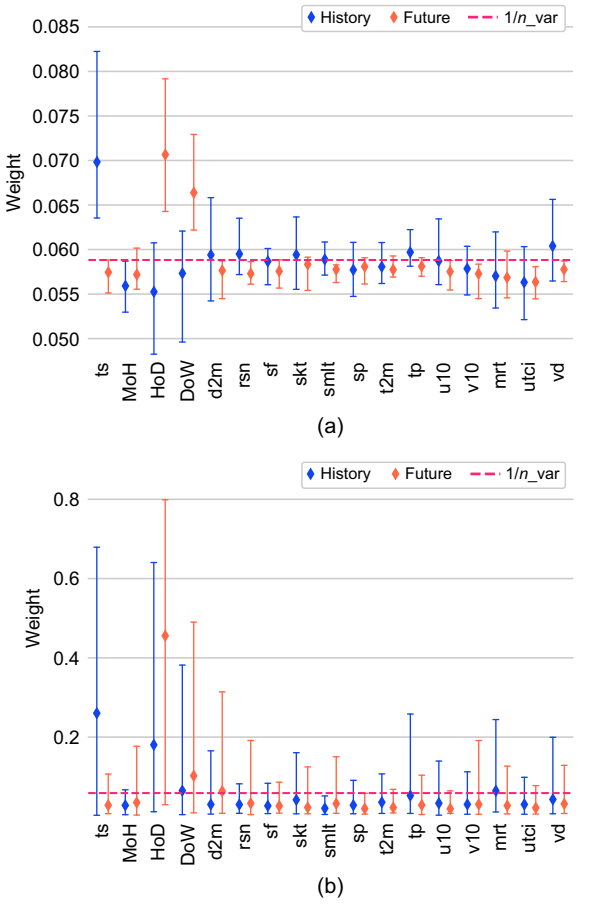

Fig. 7 Variable selection weights in Informer: (a) with TFT-GWS; (b) with TFT

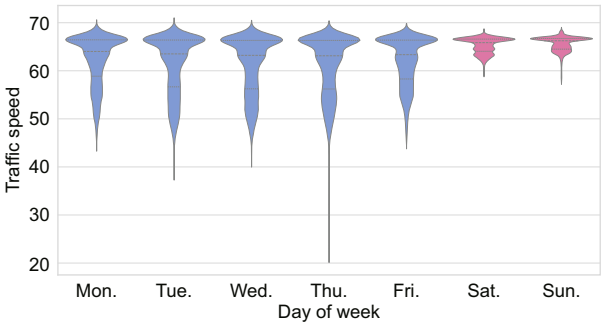

Fig. 8 Average traffic speed distribution by DoW on PEMS-BAY

our experimental results demonstrate an improved performance, longer prediction horizons are needed to fully assess the influence of external factors such as weather and special events. Furthermore, the current dataset lacks examples of extreme weather conditions, and the model architecture is limited to classical deep learning approaches.

Future work will address these limitations by focusing on more efficient data processing techniques and adaptive modeling strategies. We also plan to evaluate our method on larger datasets, such as LargeST (Liu X et al., 2023), with extended time horizons and an increased number of traffic nodes, building upon related research in traffic patterns (Shao et al., 2025; Zheng et al., 2025). Additionally, we recognize the importance of collecting data on extreme weather events to ensure the algorithm robustness in real-world scenarios.

Contributors

Yusong ZHOU conceptualized the research, performed the data collection and analysis, conducted the experiments,

and drafted the paper. Xiaoyu JIANG and Shu SUN provided valuable suggestions during the conceptualization phase. Xinmin ZHANG, Yuanqiu MO, and Zhihuan SONG performed critical review and editing of the paper, and helped with its finalization.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

All the data that support the findings of this study are available through the open-source LibCity project. Datasets are already uploaded in network disks BaiduDisk with code 1231 or Google Drive.

References

Akiba T, Sano S, Yanase T, et al., 2019. Optuna: a next-generation hyperparameter optimization framework. Proc 25th ACM SIGKDD Int Conf on Knowledge Discovery & Data Mining, p.2623-2631. https://doi.org/10.1145/3292500.3330701

Albanese D, Filosi M, Visintainer R, et al., 2013. Minerva and minepy: a C engine for the MINE suite and its R, Python and MATLAB wrappers. *Bioinformatics*, 29(3):407-408. https://doi.org/10.1093/bioinformatics/bts707

Bączek J, Zhylko D, Titericz G, et al., 2024. TSPP: a unified benchmarking tool for time-series forecasting. https://doi.org/10.48550/arXiv.2312.17100

Bai HY, Liu X, 2025. T-Graphormer: using Transformers for spatiotemporal forecasting. https://doi.org/10.48550/arXiv.2501.13274

Do LNN, Taherifar N, Vu HL, 2019. Survey of neural network-based models for short-term traffic state prediction. *WIREs Data Min Knowl Disc*, 9(1):e1285. https://doi.org/10.1002/widm.1285

Essien A, Petrounias I, Sampaio P, et al., 2018. The impact of rainfall and temperature on peak and off-peak urban traffic. Int Conf on Database and Expert Systems Applications, p.399-407. https://doi.org/10.1007/978-3-319-98812-2_36

Essien A, Petrounias I, Sampaio P, et al., 2021. A deep-learning model for urban traffic flow prediction with traffic events mined from twitter. *World Wide Web*, 24(4):1345-1368. https://doi.org/10.1007/s11280-020-00800-3

Fang Z, Long QQ, Song GJ, et al., 2021. Spatial-temporal graph ODE networks for traffic flow forecasting. Proc 27th ACM SIGKDD Conf on Knowledge Discovery & Data Mining, p.364-373. https://doi.org/10.1145/3447548.3467430

FHWA, 2024. How Do Weather Events Affect Roads? https://ops.fhwa.dot.gov/weather/q1_roadimpact.htm [Accessed on May 20, 2025].

Guo SN, Lin YF, Feng N, et al., 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. Proc 39th AAAI Conf on Artificial Intelligence, p.922-929. https://doi.org/10.1609/aaai.v33i01.3301922

Hochreiter S, Schmidhuber J, 1997. Long short-term memory. *Neur Comput*, 9(8):1735-1780. https://doi.org/10.1162/neco.1997.9.8.1735

Lee K, Eo M, Jung E, et al., 2021. Short-term traffic prediction with deep neural networks: a survey. *IEEE Access*, 9:54739-54756. https://doi.org/10.1109/ACCESS.2021.3071174

Li Y, Yu R, Shahabi C, et al., 2018. Diffusion Convolutional Recurrent Neural Network: Data-Driven Traffic Forecasting. https://openreview.net/forum?id=SJiHXGWAZ [Accessed on May 20, 2025].

Lim B, Arık SO, Loeff N, et al., 2021. Temporal fusion Transformers for interpretable multi-horizon time series forecasting. *Int J Forecast*, 37(4):1748-1764. https://doi.org/10.1016/j.ijforecast.2021.03.012

Liu J, Guan W, 2004. A summary of traffic flow forecasting methods. *J Highway Transport Res Dev*, 21(3):82-85 (in Chinese). https://doi.org/10.3969/j.issn.1002-0268.2004.03.022

Liu X, Xia YT, Liang YX, et al., 2023. LargeST: a benchmark dataset for large-scale traffic forecasting. Proc 37th Int Conf on Neural Information Processing Systems, p.75354-75371.

Ma XL, Yu HY, Wang YP, et al., 2015. Large-scale transportation network congestion evolution prediction using deep learning theory. *PLoS ONE*, 10(3):e0119044. https://doi.org/10.1371/journal.pone.0119044

Meta Research, 2019. Hydra—a Framework for Elegantly Configuring Complex Applications. https://github.com/facebookresearch/hydra [Accessed on May 20, 2025].

Nagy AM, Simon V, 2018. Survey on traffic prediction in smart cities. *Pervasive Mob Comput*, 50:148-163. https://doi.org/10.1016/j.pmcj.2018.07.004

Reshef DN, Reshef YA, Finucane HK, et al., 2011. Detecting novel associations in large data sets. *Science*, 334(6062):1518-1524. https://doi.org/10.1126/science.1205438

Shao ZZ, Wang F, Xu YJ, et al., 2025. Exploring progress in multivariate time series forecasting: comprehensive benchmarking and heterogeneity analysis. *IEEE Trans Knowl Data Eng*, 37(1):291-305. https://doi.org/10.1109/TKDE.2024.3484454

Shaygan M, Meese C, Li WX, et al., 2022. Traffic prediction using artificial intelligence: review of recent advances and emerging opportunities. *Transport Res Part C Emerg Technol*, 145:103921. https://doi.org/10.1016/j.trc.2022.103921

Song C, Lin YF, Guo SN, et al., 2020. Spatial-temporal synchronous graph convolutional networks: a new framework for spatial-temporal network data forecasting. Proc 34th AAAI Conf on Artificial Intelligence, p.914-921. https://doi.org/10.1609/aaai.v34i01.5438

Sutskever I, Vinyals O, Le QV, 2014. Sequence to sequence learning with neural networks. Proc 27th Int Conf on Neural Information Processing Systems, p.3104-3112.

Taghipour H, Parsa AB, Mohammadian AK, 2020. A dynamic approach to predict travel time in real time using data driven techniques and comprehensive data sources.

Transport Eng, 2:100025. https://doi.org/10.1016/j.treng.2020.100025

Vaswani A, Shazeer N, Parmar N, et al., 2017. Attention is all you need. Proc 31st Int Conf on Neural Information Processing Systems, p.6000-6010.

Wang JY, Jiang JW, Jiang WJ, et al., 2021. LibCity: an open library for traffic prediction. Proc 29th Int Conf on Advances in Geographic Information Systems, p.145-148. https://doi.org/10.1145/3474717.3483923

Wu HX, Hu TG, Liu Y, et al., 2023. TimesNet: temporal 2D-variation modeling for general time series analysis. https://doi.org/10.48550/arXiv.2210.02186

Wu ZH, Pan SR, Long GD, et al., 2019. Graph WaveNet for deep spatial-temporal graph modeling. Proc 28th Int Joint Conf on Artificial Intelligence, p.1907-1913. https://doi.org/10.24963/IJCAI.2019/264

Yang XX, Zou YJ, Tang JJ, et al., 2020. Evaluation of short-term freeway speed prediction based on periodic analysis using statistical models and machine learning models. *J Adv Transport*, 2020(1):9628957. https://doi.org/10.1155/2020/9628957

Yu B, Yin HT, Zhu ZX, 2018. Spatio-temporal graph convolutional networks: a deep learning framework for traffic forecasting. Proc 27th Int Joint Conf on Artificial Intelligence, p.3634-3640. https://doi.org/10.24963/ijcai.2018/505

Zhao L, Song YJ, Zhang C, et al., 2020. T-GCN: a temporal graph convolutional network for traffic prediction. *IEEE Trans Intell Transport Syst*, 21(9):3848-3858. https://doi.org/10.1109/TITS.2019.2935152

Zheng X, Bagloee SA, Sarvi M, 2025. A comparative study of deep learning and ensemble learning to extend the horizon of traffic forecasting. https://doi.org/10.48550/arXiv.2504.21358

Zhou HY, Zhang SH, Peng JQ, et al., 2021. Informer: beyond efficient Transformer for long sequence time-series forecasting. Proc 39th AAAI Conf on Artificial Intelligence, p.11106-11115. https://doi.org/10.1609/aaai.v35i12.17325

Zhu L, Yu FR, Wang YG, et al., 2019. Big data analytics in intelligent transportation systems: a survey. *IEEE Trans Intell Trans Syst*, 20(1):383-398. https://doi.org/10.1109/TITS.2018.2815678

List of supplementary materials