mathematics

MDPI

Article

Mathematical Modeling of Transit Network Spatiotemporal Dynamics: A Data-Efficient Graph-BiLSTM Architecture for Segment-Level Travel Time Forecasting

Aigerim Mansurova [1,*], Aigul Adamova [2,*], Daniyar Rakhymzhanov [3] and Didar Yedilkhan [4]

[1] Big Data and Blockchain Technologies Research Center, Astana IT University, Astana 010000, Kazakhstan
[2] School of Software Engineering, Astana IT University, Astana 010000, Kazakhstan
[3] School of Artificial Intelligence and Data Science, Astana IT University, Astana 010000, Kazakhstan; d.rahimzhanov@astanait.edu.kz
[4] Smart City Research Center, Astana IT University, Astana 010000, Kazakhstan; d.yedilkhan@astanait.edu.kz
* Correspondence: a.mansurova@astanait.edu.kz (A.M.); aigul.adamova@astanait.edu.kz (A.A.)

Abstract

This study addresses the challenge of accurate short-term bus travel time prediction in data-constrained urban environments. While many existing models rely on extensive external data, we propose a data-efficient spatiotemporal framework that integrates sequential deep learning with multi-relational graph modeling using only standard operational transit data. The approach is implemented through a multi-relational graph spatio-temporal bidirectional LSTM (MRG-ST-BiLSTM) model, which captures temporal dependencies while preserving the physical topology of the road network. A real-world case study was conducted using GTFS-based data from bus operations in the Greater Sydney Area, covering five routes and more than four million stop-level records. In the real road network test, numerical experiments demonstrated that the proposed model achieved improved prediction accuracy compared to baseline methods. Specifically, it achieved an MAE of 15.82 s and RMSE of 31.38 s, outperforming the conventional LSTM (MAE 26.70, RMSE 45.61), Hybrid-BiLSTM (MAE 17.24, RMSE 32.73) and GCN-LSTM (MAE 17.27, RMSE 32.61). The proposed framework provides a scalable and interpretable practical solution for transit agencies.

Keywords: bus travel time prediction; graph-enhanced BiLSTM; intelligent transportation system (ITS); spatio-temporal public transit modeling; data-efficient deep learning

MSC: 68T07; 90B06; 05C90; 62M10

Check for updates

Academic Editor: Noureddine Lakouari

Received: 1 April 2026
Revised: 21 April 2026
Accepted: 24 April 2026
Published: 29 April 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Within the framework of intelligent transportation systems (ITSs), providing accurate bus travel and arrival time estimates is no longer just a convenience for passengers, but a prerequisite for dynamic fleet management and congestion control. While short-term prediction remains a critical challenge for modern transportation management, the recent proliferation of complex deep learning models has introduced new handles. Many of these architectures function as "black boxes" dependent on high-resolution data that municipal transport authorities often lack [1]. Moreover, existing methods frequently overlook the physical topology of bus networks, necessitating a shift toward data-efficient models that maintain high accuracy using only standard operational records while providing the interpretability required for urban planning.

Recent advances in graph neural networks (GNNs) have fundamentally reshaped spatiotemporal forecasting. For instance, while some studies demonstrate the effectiveness of multi-task dynamic attention in aviation environments, others emphasize integrating external factors through specialized attention modules [2,3]. In the context of public transit, graph attention networks (GATs) have been utilized to model congestion propagation between stops, significantly outperforming classical sequence models [4]. Further research has highlighted the importance of segment-level granularity and decoupled interactive learning to improve forecast stability [5,6].

Despite these contributions, existing frameworks typically rely on computationally prohibitive architectures or multi-source data that are unavailable in many municipal settings. Crucially, many models fail to exploit the multi-relational nature of bus networks, such as the distinction between physical adjacency and shared route overlaps. This study addresses these gaps by proposing a data-efficient spatiotemporal model specifically designed for short-term forecasting using only standard operational transport records (GTFS). By preserving the physical network topology, the proposed approach aims to bridge the gap between complex algorithmic outputs and the functional requirements of urban transport planning.

The primary objective of this paper was to develop a spatiotemporal modeling framework for short-term bus travel time prediction that integrates sequential deep learning with multi-relational graph modeling, relying exclusively on operational transit data. To achieve this, the study addresses the following research questions:

RQ1. To what extent can bus segment travel times be accurately predicted using only GTFS-derived operational records in the absence of external contextual variables?

RQ2. How does the integration of multi-relational spatial dependencies between route segments affect prediction performance compared to purely temporal models?

The primary contributions of this research include the following:

- We formalize bus travel time forecasting as a segment-level spatiotemporal task that explicitly integrates temporal dynamics with the physical topology of the transit network.
- Unlike existing models that rely on external variables, our framework utilizes only standard operational transit data. This approach significantly enhances the LSTM's scalability and its potential for deployment in resource-constrained environments.
- We introduce a lightweight graph-enhanced hybrid BiLSTM architecture. This model uniquely combines categorical transit feature embedding with BiLSTM-based temporal modeling and graph-based spatial aggregation, ensuring both predictive precision and operational interpretability.
- We empirically validated the framework using real-world bus operational data.

The remainder of this paper is structured as follows. Section 2 reviews the relevant literature on spatiotemporal forecasting and graph-based transit modeling. Section 3 details the proposed graph-enhanced hybrid BiLSTM architecture and the data preprocessing workflow. In Section 4, we present the experimental results and a comparative performance analysis. Finally, Section 5 discusses the implications of our findings and outlines potential avenues for future work.

2. Literature Review

Numerous methods have been proposed for the bus travel time prediction, incorporating a range of techniques and algorithms. Early studies relied on statistical and traditional machine learning methods such as historical averages [7–9], Kalman filter (KF) [10–14], ARIMA [15–17], support vector regression (SVR) [18–22], and artificial neural networks (ANNs) [20]. However, these models typically assume stable travel patterns and therefore struggle to capture the complex and highly dynamic nature of urban traffic conditions.

With the advancement of deep learning, researchers have demonstrated that bus travel time prediction can be significantly improved by using deep learning architectures such as long short-term memory (LSTM) networks, which effectively capture temporal dependencies [23–25]. Compared to traditional machine learning models, LSTMs have shown superior performance in modeling sequential data, making them particularly suitable for bus arrival time estimation [25,26]. For instance, the authors of [25] demonstrated that an LSTM model outperformed baseline approaches such as ANN, SVR, ARIMA, and historical averages in predicting transit bus arrival time irregularities using GPS and weather data from the Toronto Transit Commission. Moreover, Hou and Edara [26] predicted link travel time using an LSTM model and demonstrated its superiority over a convolutional neural network (CNN) model. However, despite their advantages, standalone LSTM models often struggle with capturing spatial dependencies.

A number of LSTM-based studies also rely on detailed vehicle-level operational datasets collected through onboard sensors and telematics systems [20,27,28]. These datasets typically include operational parameters such as vehicle speed, acceleration patterns, braking behavior, engine diagnostics, fuel consumption, and passenger occupancy levels. Such data are obtained through onboard vehicle sensors, telematics platforms, or automated fare collection systems. For example, Yu et al. [20] collected operational data including bus running time, arrival time, and timestamps from automatic toll tag readers to train and evaluate several predictive models. Similarly, other studies have used large-scale operational datasets containing numerous attributes describing driving states, mechanical conditions, and GPS locations of buses. While these data-rich approaches can significantly improve the prediction accuracy, they require specialized sensing infrastructure and extensive data collection systems that are not available in many cities.

Several studies have attempted to improve the prediction performance by incorporating additional operational and passenger-related information. In particular, some approaches integrate passenger demand data to implicitly account for dwell time variations and link travel time delays caused by passenger boarding and alighting activities [23,26,29]. Other studies rely on the General Transit Feed Specification (GTFS) data [30–33]. Compared with estimated operational data, GTFS allows researchers to model real-world transit operations more accurately. A widely adopted approach is to segment bus routes according to bus stop locations and estimate the travel time for each segment separately. For example, Gal et al. [31] partitioned bus routes into segments between stops and estimated the travel time using data from multiple bus lines traversing the same segment. Wu et al. [32] further incorporated attention-based convolutional LSTM architectures to model travel time across route segments using spatio-temporal features derived from transit operations. Ratneswaran and Thayasivam [33] proposed a multi-model ensemble framework designed to improve travel time prediction under heterogeneous traffic conditions. Although such approaches provide improved spatial granularity, they often do not explicitly consider passenger waiting times or delays associated with route transfers.

To better capture spatial dependencies between bus stops and route segments, more recent studies have explored hybrid deep learning architectures that combine LSTM models with convolutional or graph-based neural networks. Convolutional LSTM models have been proposed to jointly model spatial correlations between route segments and the temporal dynamics of travel time sequences [34]. For instance, Lee et al. [35] developed a Geo-convLSTM model that integrates convolutional layers for spatial feature extraction with LSTM layers for temporal modeling. The model incorporates multiple datasets including smart card passenger transactions, road speed data, weather information, and operational variables. Liu et al. [36] proposed an ANN–LSTM architecture that extracts spatial features from route segments, while LSTM layers capture temporal travel time

variations. Similarly, Jiang et al. [37] introduced a deep learning framework capable of handling incomplete datasets by combining multiple data sources.

More recently, graph-based neural networks have been introduced to represent transportation systems as structured networks where nodes correspond to bus stops or road segments and edges represent the connectivity between them. These approaches allow models to capture spatial interactions between neighboring locations more effectively. For example, Li et al. [38] proposed a sequence and network embedding approach for bus arrival time prediction using GPS trajectory data, learning both temporal and network-level representations of bus movement patterns. Huang et al. [39] introduced the GT-TTE framework, which models trajectories as graphs and uses advanced neural architectures to capture spatial and temporal relationships within trajectory sequences. Other recent frameworks integrate attention-based recurrent neural networks and graph convolutional networks to model complex interactions across multiple routes and stops [40,41]. To provide a structured overview of the current landscape, Table 1 summarizes key recent developments in the field, contrasting their methodological focus with the requirements of operational efficiency.

Table 1. Summary of related work.

Ref.	Methodology	Data Requirements	Spatial Awareness	Key Contribution
[35]	CNN + LSTM	Rich features set beyond AVL: road speed features, driver identifiers, weather variables.	Route-local spatial modeling via geo-convolution over adjacent stops, augmented with stop-to-stop distance.	Proposes a stop-sequence convolution approach intended to avoid grid-size artifacts.
[37]	LSTM/GRU/Bi-LSTM	Hybrid smart-card (ESC) and GPS streams to derive arrival/departure times.	Sequence-only: stop-to-stop time series along a route (no explicit network/graph).	Hybrid ESC + GPS improves error versus single-source baselines.
[38]	SGE-net: sequence embedding (Echo State Network + Auto-Encoder + k-means) + network embedding + fusion via parallel GRU networks.	GPS trajectories only.	Graph/topology-explicit: node2vec over stops.	Provides a GPS-only, data-accessible framework.
[39]	GT-TTE a graph-transformer.	GPS points with temporal ordering.	Graph/topology-explicit: graph over trajectory nodes and region adjacency.	Addresses the challenge of extracting regional spatial information from trajectory formats. Reports improvement compared with other graph transformers and statistical models.
[33]	Multi-model stacked generalization ensemble: ConvLSTM + XGBoost.	Standard operational data, weather features, smart-card/ESC, APC/passenger-flow, and traffic-flow sensor inputs.	No explicit network/graph.	Improves next-stop arrival-time accuracy in heterogeneous traffic by stacking ConvLSTM and XGBoost over a running/dwell decomposition.
[30]	Hybrid-BiLSTM	Standard operational only	Sequence-only (route-level segmentation), no explicit network/graph	LSTM models using operational variables can improve bus arrival time prediction without complex external features.

Table 1. *Cont.*

Ref.	Methodology	Data Requirements	Spatial Awareness	Key Contribution
[40]	PMLNet: LSTM + MDARN calibrates via four real-time traffic impact factors.	Requires both operational and public databases: local (AVL data, traffic flow) and macro factors (weather/temperature/air quality).	No explicit network/graph.	Introduces attention-based recurrent architecture to model interactions across multiple bus routes.
[41]	FEN-MRMGCN	Comprehensive multi-source dataset including passenger flow, traffic conditions, weather, bus IDs, inter-stop distances, and arrival times.	Multi-relational graph convolution.	Multi-relational Graph Convolutional Network (GCN) capturing inter-route spatial dependencies with temporal models, improving prediction accuracy.
This Study	Graph Hybrid BiLSTM	Standard operational only	Route-level segmentation with topology-preserving	Data efficient, interpretable, scalable

This research contributes to the existing body of literature by introducing a mathematically consistent framework that embeds the physical topology of the transport network directly into a spatiotemporal learning architecture. Specifically, we formalized bus travel time prediction as a segment-level task where spatial dependencies are resolved through graph-enhanced aggregation and temporal dynamics are captured via a lightweight Hybrid BiLSTM. Unlike computationally intensive 'black-box' models that rely on external variables, the proposed approach utilizes only standard operational records. By prioritizing data efficiency and structural stability, this study provides a unified analytical foundation for forecasting that effectively bridges the gap between high-level architectural complexity and the functional requirements of real-time transit management.

3. Materials and Methods

3.1. Problem Description

The study addresses the problem of short-term bus travel time prediction, where bus trips are modeled as ordered sequences of segments. A segment refers to the directional road section between two consecutive bus stops along a route. Let a trip consist of an ordered sequence of stops $\ldots, i, i+1, \ldots$ For each segment t, connecting stop $i-1$ to stop i, let Arr_i denote the arrival time at stop i and Dep_{i-1} denote the departure time from stop $i-1$.

The run time of segment t, representing the in-motion travel time between two consecutive stops, is defined as:

$$\text{run_time}_t = \text{Arr}_i - \text{Dep}_{i-1}. \quad (1)$$

The dwell time associated with segment t is defined at the upstream stop as:

$$\text{dwell_time}_t = \text{Dep}_{i-1} - \text{Arr}_{i-1}. \quad (2)$$

representing the time the vehicle remains stopped at stop $i-1$ before departing toward stop i.

Accordingly, the target variable travel time t is defined as:

$$\text{y}_t = \text{travel_time}_t = \text{run_time}_t + \text{dwell_time}_t. \quad (3)$$

Given historical observations from previous segments within a trip, the objective is to predict y_t for the current target segment. Figure 1 illustrates the formulation of the bus travel time prediction problem at the stop-to-stop segment level.

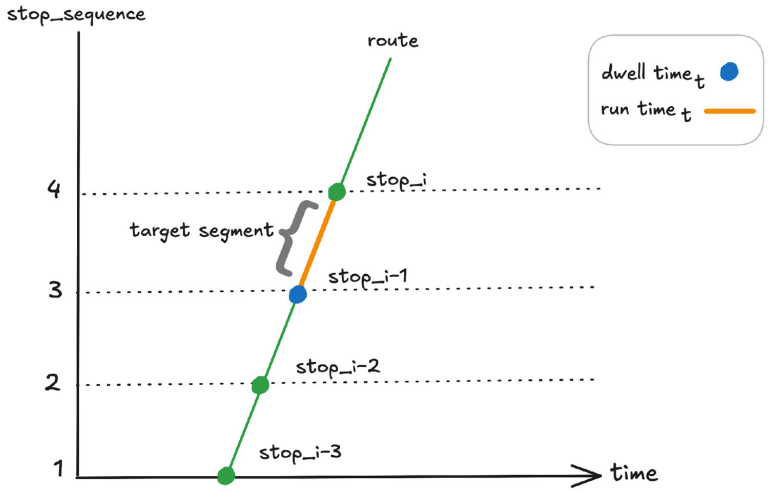


Figure 1. Formulation of the bus travel time prediction problem at the stop-to-stop segment level.

3.2. Data Description

3.2.1. Bus Arrival and Departure Time Dataset for Greater Sydney

The proposed model was evaluated using a real-world public transport dataset derived from the study [42]. The dataset is based on GTFS-Realtime Trip Updates provided by Transport for New South Wales (TfNSW) and contains processed stop-level arrival and departure time information for bus services operating in the Greater Sydney Area.

For this study, a subset of the original dataset was extracted covering 5 routes (2459_423, 2459_428, 2435_802, 2435_804, and 2435_806) over the period from 1 January 2021 to 1 May 2021. Selected routes contained the largest volume of observations and the highest number of shared intersection stops, as illustrated by the stop-level network graph in Figure 2. The dataset comprises 4,200,641 stop-level trip update records derived from 5160 vehicles. The distributions of run time between consecutive stops and dwell time at stops are illustrated in Figure 3.

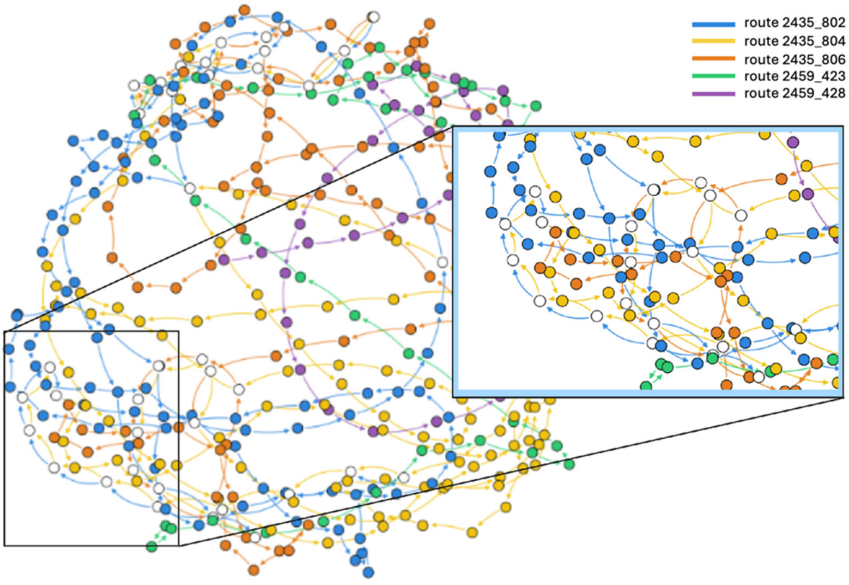
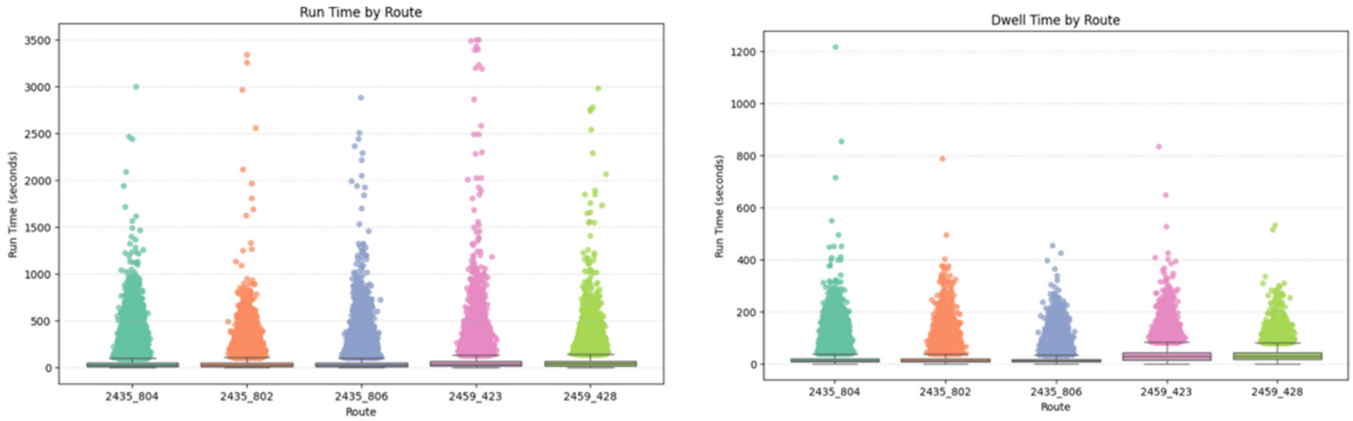


Figure 2. Stop-level network graph of the selected bus routes (white nodes indicate intersection stops).



(**a**) (**b**)

Figure 3. Distributions of (**a**) run time between consecutive stops and (**b**) dwell time at stops across the selected bus routes.

3.2.2. Data Preprocessing

To transform the stop-level GTFS-Realtime records into a segment-level dataset suitable for our predictive modeling, a multistage preprocessing procedure was applied. First, only essential attributes describing trips, vehicles, stop sequences, and timestamps were retained. All time fields were converted to datetime format, and records with missing or invalid timestamps were removed to ensure temporal consistency. It should be noted that the trip_id field in the original dataset corresponds to a scheduled trip identifier and does not uniquely identify individual executed trips. Therefore, to distinguish real executions of scheduled trips, a unique trip instance identifier was created by combining vehicle ID and trip start time. Records were then chronologically ordered within each trip, and consecutive stops were paired to form stop-to-stop segments. For each segment, dwell time at the upstream stop, run time to the downstream stop, and total travel time were computed according to Equations (2)–(4).

Temporal identifiers were additionally introduced to capture systematic variations in bus operations across different days of the week. In particular, a day-of-week indicator was included as a categorical feature to account for differences between weekday and weekend traffic conditions. As illustrated in Figure 4, the average segment travel times exhibited noticeable variation across days, especially during peak hours, reflecting differences in passenger demand and traffic congestion patterns. Incorporating this temporal context allows the model to better capture recurring operational patterns in the public transport system.

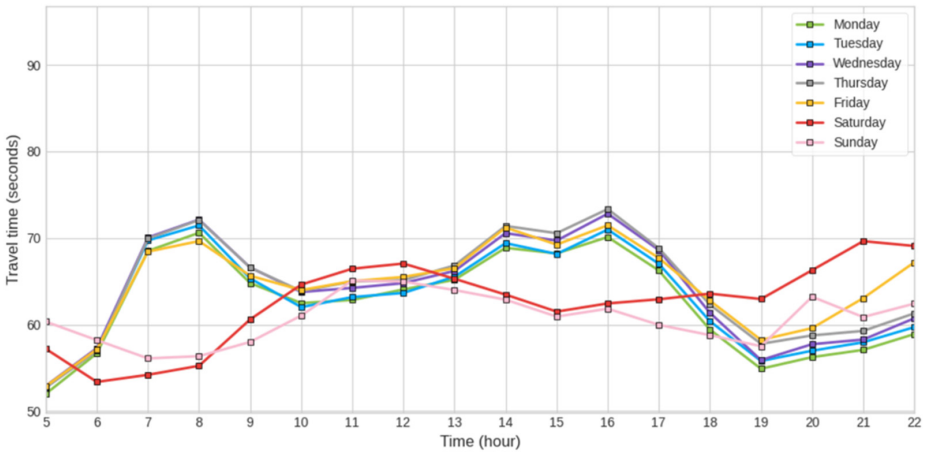


Figure 4. Average bus travel time (seconds) by hour of day across weekdays and weekends.

Finally, unrealistic observations were filtered to remove noise and errors. The overall data preprocessing procedure is summarized in Algorithm 1, while Table 2 lists the corresponding explanations of variables and symbols.

Algorithm 1. Construction of the segment-level dataset from stop-level GTFS records

Input: Raw stop-level dataset D_{raw} containing route identifiers, scheduled trip identifiers, vehicle identifiers, stop sequence, stop identifier, arrival time, departure time, trip start time, and delay-related fields.
Output: Clean segment-level dataset D_{seg} for travel time prediction.

// Preprocessing of stop-level records:
Select essential attributes from D_{raw}:
(route_id, trip_id, vehicle_id, stop_sequence, stop_id, Arr_i, Dep_i, trip_start_time, arrival_delay_i, departure_delay_i).
Convert all temporal fields to datetime format.
Remove records with missing, duplicated, or invalid timestamps.
Construct a unique executed trip identifier:
trip_instance_id = vehicle_id $\oplus$ trip_start_time.
Sort all records by (trip_instance_id, stop_sequence).
// Construction of stop-to-stop segments:
for each trip_instance_id k in D_raw do
    for each consecutive stop pair (i, i + 1) within trip k do
        Define start_point = stop_id_i;
        Define end_point = stop_id_{i + 1};
        Define segment index = position of pair (i, i + 1) in trip k;
        Compute dwell time at the upstream stop as in Equation (2);
        Compute run time between consecutive stops as in Equation (1);
        Compute total segment travel time as in Equation (3);
        Extract service date:
            date_i = date(trip_start_time);
        Extract day-of-week:
            day_of_week_i = weekday(date_i);
        Extract trip start time only:
            trip_start_time_only_i = time(trip_start_time);
        Store one segment-level observation:
            (trip_instance_id, route_id, segment, date_i, day_of_week_i, trip_start_time_only_i, start_point, end_point, dwell_time_i, run_time_i, travel_time_i).
    end
end
// Cleaning of segment-level observations:
Remove observations with:
    dwell_time_i < 0,
    run_time_i $\leq$ 0,
    travel_time_i $\leq$ 0.
Remove extreme outliers:
    travel_time_i > Q_99 (travel_time) predefined upper bound.
Return the resulting clean segment-level dataset D_seg.

The resulting dataset comprises 3,655,384 records derived from 70,275 trips across five routes. A detailed description of all variables included in the dataset is provided in Table 3.

Table 2. The explanation for Algorithm 1 variables and functions.

Symbol/Variable	Explanation
D_raw	Raw stop-level dataset
D_seg	Final clean segment-level dataset
trip_id	Scheduled trip identifier
vehicle_id	Unique vehicle identifier
trip_start_time	Start time of the executed trip
trip_instance_id	Unique identifier of an executed trip, formed by concatenating vehicle ID and trip start time
i	Stop index within a trip
t	Segment index between two consecutive stops
stop_id_i	Identifier of stop (i)
Arr_i	Arrival time at stop (i)
Dep_i	Departure time from stop (i)
arrival_delay_i	Arrival delay recorded at stop (i), if available in the original data
departure_delay_i	Departure delay recorded at stop (i), if available in the original data
start_point	Upstream stop of a segment
end_point	Downstream stop of a segment
dwell_time_t	Time spent at the upstream stop before departure
run_time_t	In-motion time between two consecutive stops
travel_time_t	Total segment travel time, equal to dwell time plus run time
date_t	Service date of the trip
day_of_week_t	Day-of-week category derived from service date
trip_start_time_only_t	Time-of-day component of the trip start time
$\oplus$	Concatenation operator
Q_99	99th percentile

Table 3. Description of variables in the segment-level dataset.

Variable	Description
trip_id	Unique identifier of the trip.
route_id	Identifier of the bus route to which the trip belongs.
segment	Stop-to-stop segment number within the trip sequence.
date	Service date when the trip was executed.
day_of_week	Categorical variable indicating the day of the week (Monday–Sunday), introduced to capture weekly variations in traffic conditions and passenger demand.
trip_start_time_only	Departure time of the trip from the first stop, representing the temporal context of the trip.
start_point	Identifier of the starting stop of the segment.
end_point	Identifier of the next stop defining the end of the segment.
run_time_in_seconds	Time required for the bus to travel between two consecutive stops.
dwell_time_in_seconds	Time spent at the stop for passenger boarding and alighting.
travel_time	Total segment travel time computed as the sum of run time and dwell time.

3.3. Proposed Predictive Modeling Framework

The proposed framework formulates bus segment travel time prediction as a graph-enhanced sequence-to-one regression problem. For each target segment t, the model estimates its travel time y_t using:

- a fixed-length historical window of preceding segments within the same trip;
- categorical and continuous operational features describing each segment;
- cyclical temporal context;
- and graph-based stop representations learned from the underlying transport network.

Formally, the prediction task can be written as

$$\hat{y}_t = f_\theta\,(W_t,\ x_t,\ r_t;\ A,\ B), \tag{4}$$

where W_t denotes the historical input window associated with target segment t, x_t denotes temporal features, r_t denotes the route context, A denotes the co-route similarity graph, B denotes the sequential stop adjacency graph, and f_θ is the proposed graph-enhanced BiLSTM model parameterized by θ. The overall architecture of the framework is illustrated in Figure 5.

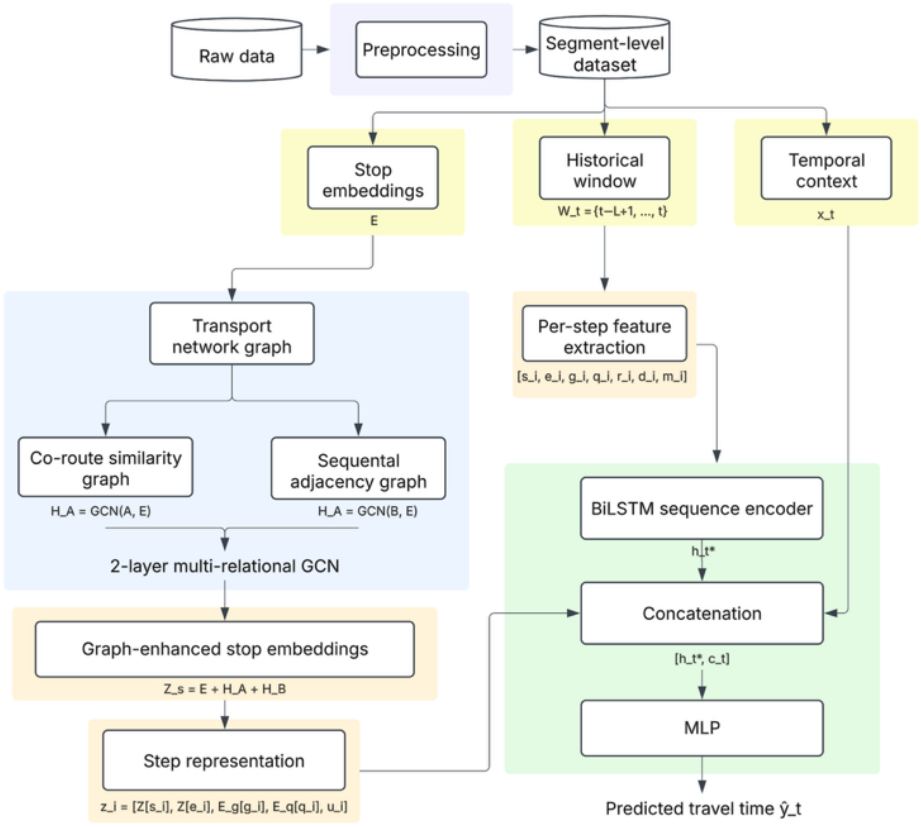


Figure 5. General framework of the proposed MRG-ST-BiLSTM model.

3.3.1. Input Features Representation

For each target segment t, a fixed-length historical window of preceding segments is used to construct the model input:

$$W_t = \{t - L + 1,\ \ldots,\ t\}, \tag{5}$$

where L denotes the window size.

Each segment $i \in W_t$ is described by the following categorical segment level variables:

- start stop identifier as $s_i \in$ S;
- end stop identifier as $e_i \in$ S;
- segment identifier as $g_i \in$ G (ordered start and end stops pair);
- position index within the trip q_i;
- route identifier $r_i \in$ R.

All categorical variables are mapped into integer indices using label encoders and embedded into dense vector representations using trainable embedding matrices.

In addition to categorical variables, each segment i is associated with continuous operational variables:

$$r_i^{(run)} = \text{run_time}_i, \tag{6}$$

$$d_i^{(dwell)} = dwell_time_i. \tag{7}$$

To prevent information leakage during prediction, the current-step run and dwell times are masked for the segment being predicted. A binary indicator variable is introduced to identify the target position:

$$m_i = \begin{cases} 1, & if\ i = t, \\ 0, & otherwise. \end{cases} \tag{8}$$

The masked $r_i^{(run)}$ *and* $d_i^{(dwell)}$ are then defined as:

$$\tilde{r_i} = \begin{cases} 0, & if\ i = t, \\ r_i^{(run)}, & otherwise. \end{cases} \tag{9}$$

$$\tilde{d_i} = \begin{cases} 0, & if\ i = t, \\ d_i^{(dwell)}, & otherwise. \end{cases} \tag{10}$$

Thus, each step i has a continuous feature vector

$$\mathrm{u}_i = \left[\tilde{r_i}, \quad \tilde{d_i}, \quad m_i\right] \in R^3. \tag{11}$$

Temporal data are encoded using cyclical transformations of time:

$$\mathrm{x}_t = [sin(2\pi h/24),\ cos(2\pi h/24),\ sin(2\pi d/7),\ cos(2\pi d/7)], \tag{12}$$

where h and d denote the hour of day and day of week, respectively. This encoding preserves the periodic structure of temporal variables and avoids artificial discontinuities between adjacent cyclical values.

3.3.2. Graph-Enhanced Stop Representation

Travel time depends not only on temporal dynamics within a trip, but also on the topology and structural organization of the transport network. To incorporate these spatial dependencies, we introduced a network-aware stop representation module that leverages structural relationships between stops to learn enriched embeddings prior to sequence modeling.

Let S denote the set of bus stops, with $\mid S \mid = N$. Each stop s $\in$ S is initially represented by a learnable embedding vector, and collecting all stop embeddings yields the matrix:

$$E \in R^{\mathrm{N} \times \mathrm{d_s}}.$$

where $\mathrm{d_s}$ denotes the stop embedding dimension.

However, standalone embeddings cannot capture network connectivity. Therefore, we constructed two complementary graphs from the training data: (i) weighted co-route similarity graph A, and (ii) sequential adjacency graph B.

Co-route similarity graph A captures the functional similarity between stops. Stops frequently served by the same routes tend to share demand patterns and traffic conditions even if they are not consecutive along a route. A stop-route incidence matrix $M \in R^{N \times |R|}$ is constructed with entries:

$$M_{s,\,r} = \begin{cases} 1, & if\ stop\ s\ appears\ on\ route\ r, \\ 0, & otherwise. \end{cases} \tag{13}$$

The stop similarity matrix is then computed as:

$$S = M\,M^{T}. \tag{14}$$

where S_{ij} indicates the number of shared routes between stops i and j. The matrix is subsequently symmetrized to obtain an undirected graph.

To preserve sparsity and avoid constructing noisy connections, only the top-k most similar neighbors per stop are retained. Specifically, for each node i, the k stops with the largest values of S_{ij} are selected, and the corresponding edges are preserved with weights equal to the shared route counts. Let $N_k(i)$ denote the set of top-k neighbors of stop i according to S_{ij}.

The final adjacency matrix A is then symmetrized and used as a weighted graph representing route-level similarity between stops.

$$A_{ij} = \begin{cases} 1, & if\ i \in N_k(i)\ , \\ 0, & otherwise. \end{cases} \tag{15}$$

Sequential adjacency graph B models physical route continuity. Stops that appear consecutively within the same route are connected. Formally, for each route r, if stop i is followed by stop j, the entries B_{ij} and B_{ji} are set to 1:

$$B_{ij} = \begin{cases} 1, & if\ stops\ i\ and\ j\ are\ consecutive\ on\ a\ route\ , \\ 0, & otherwise. \end{cases} \tag{16}$$

The resulting matrix $B \in R^{N \times N}$ therefore captures the local structural topology of the transportation network by linking neighboring stops along routes.

To illustrate the construction of the proposed graphs, consider a simplified example consisting of three bus routes and seven stops, as in Figure 6. The routes are defined as:

- R_1: A → B → C → D;
- R_2: E → F → C → D;
- R_3: A → B → G → D.

Several stops are shared across multiple routes. For instance, stops A and B appear in routes R1 and R3, stop C appears in routes R1 and R2, and stop D appears in all three routes. Other stops (E, F, and G) are unique to a single route.

Based on these routes, two graphs are constructed. In the co-route similarity graph A (see Figure 7a) the pair (B, C) appears in routes R1 and R2 and therefore receives weight 2, while the pair (C, D) appears in routes R1 and R3 and receives weight 2. In the sequential adjacency graph B (see Figure 7b), edges A–B, B–C, and C–D are created from route R1, while edges E–F and F–C originate from route R2, and edges B–G and G–D originate from

route R3. This graph therefore captures the local topology of bus routes. All other pairs that share only one route receive weight 1.

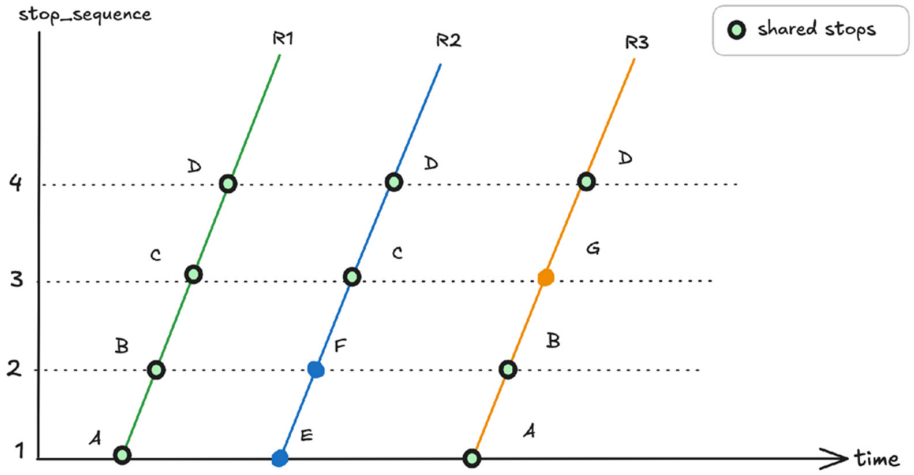
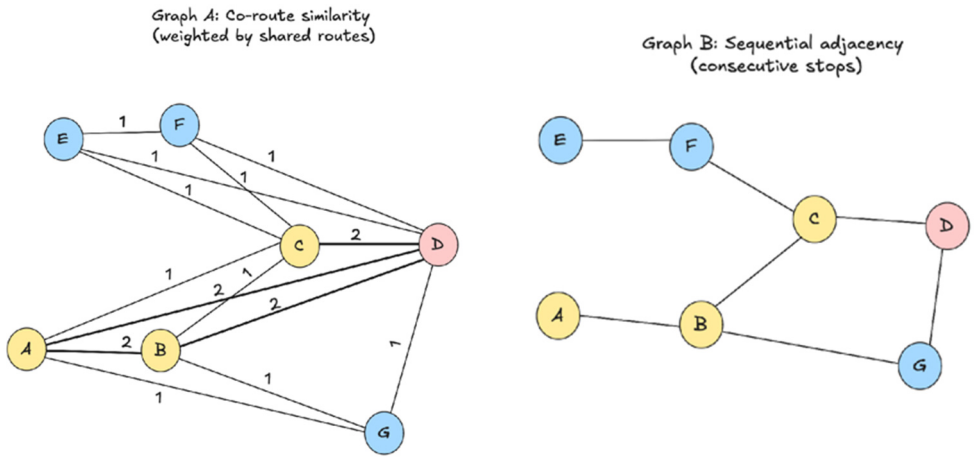


Figure 6. Example of bus routes with shared stops used to illustrate graph construction.



(**a**) (**b**)

Figure 7. Graph representations derived from the example routes: (**a**) weighted co-route similarity graph A; (**b**) sequential adjacency graph B. Nodes that appear in all three routes are highlighted in red, nodes shared by two routes are shown in orange, and nodes unique to a single route are shown in blue.

The adjacency matrices corresponding to the example routes are:

$$A = \begin{bmatrix} 0 & 2 & 1 & 2 & 0 & 0 & 1 \\ 2 & 0 & 1 & 2 & 0 & 0 & 1 \\ 1 & 1 & 0 & 2 & 1 & 1 & 0 \\ 2 & 2 & 2 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Before graph convolution, both adjacency matrices are then symmetrically normalized as:

$$\begin{aligned} \hat{A} &= D_A^{-1/2}(A + I)D_A^{-1/2}, \\ \hat{B} &= D_B^{-1/2}(B + I)D_B^{-1/2}. \end{aligned} \tag{17}$$

where D_A and D_B denote the corresponding degree matrices and I is the identity matrix. I is added to the adjacency matrix to introduce self-loops, allowing each node to preserve its own information during graph convolution.

A two-layer GCN is then applied to both relations in order to propagate information between connected stops. The same weight parameters are shared across the two relation types to reduce model complexity. The propagation process can be written as:

$$H_A = \hat{A}\,\sigma\left(\hat{A}\,E_s W_1\right)W_2, \tag{18}$$

$$H_B = \hat{B}\,\sigma\left(\hat{B}\,E_s W_1\right)W_2, \tag{19}$$

where $E_s \in R^{N\times d}$ represents the initial stop embedding matrix, W_1 and W_2 are learnable weight matrices, and $\sigma(\cdot)$ denotes the nonlinear activation function.

Finally, the graph-enhanced stop embedding matrix is obtained through residual fusion:

$$\mathrm{Z}_s = E + \mathrm{H}_A + \mathrm{H}_B.$$

This design allows the model to jointly capture local route continuity, global route similarity, and residual identity information.

3.3.3. Sequential Modeling Using BiLSTM

After obtaining graph-enhanced stop embeddings, the prediction task is formulated as a sequence-to-one regression problem. A long short-term memory (LSTM) network is an advanced variant of the recurrent neural network (RNN), specifically designed to address the challenges associated with long-term dependency learning, particularly the issues of gradient vanishing and explosion encountered in traditional RNNs.

The inner structure of an LSTM cell is shown in Figure 8. The memory cells facilitate the selective retention and removal of information through a gating mechanism composed of three key components: the input gate (i_t), forget gate (f_t), and output gate (o_t). The forget gate regulates the retention of previous cell state information (c_{t-1}), ensuring that only relevant data are preserved. The input gate determines the new information to be integrated into the cell state, while the output gate governs the information passed to the next time step or network layer. This gated structure allows the model to effectively learn temporal dependencies in transportation time-series data, where current travel conditions are influenced by previous segment dynamics.

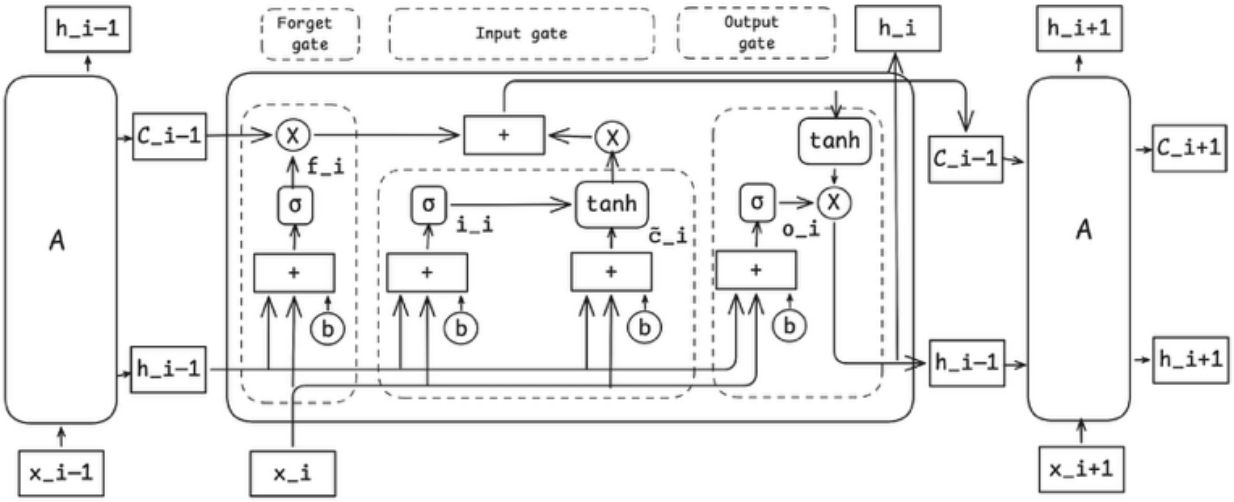


Figure 8. The inner structure of an LSTM cell.

Mathematically, given the input at time step i, *denoted by* x_t, the hidden state from the previous time step h_{i-1}, and the previous cell state (c_{i-1}), the update equations for an LSTM cell are defined as follows:

$$f_i = \sigma\Big(W_f x_i + U_f h_{i-1} + b_f\Big), \tag{20}$$

$$i_i = \sigma(W_i x_i + U_i h_{i-1} + b_i), \tag{21}$$

$$o_i = \sigma(W_o x_i + U_o h_{i-1} + b_o), \tag{22}$$

$$\widetilde{c}_i = \tanh(W_c x_i + U_c h_{i-1} + b_c), \tag{23}$$

$$c_i = f_i \odot c_{i-1} + i_i \odot \widetilde{c}_i\,, \tag{24}$$

$$h_i = o_i \odot tanh\,(c_i)\,, \tag{25}$$

where σ denotes the sigmoid activation function, tanh denotes the hyperbolic tangent function, W and U are weight matrices corresponding to the input and hidden state, respectively, b represents the bias vectors, which are learned during the training process, and $\odot$ denotes the element-wise (Hadamard) multiplication.

To better exploit contextual dependencies from both earlier and later positions within the historical window, this study employed a bidirectional long short-term memory (BiLSTM) network. Unlike standard LSTMs that operate solely in a forward temporal direction, BiLSTMs utilize two parallel LSTM layers (see Figure 9): one processes the input from past to future (forward direction), while the other operates from future to past (backward direction). This bidirectional structure allows the model to utilize contextual information from both preceding and succeeding bus stops, thereby enhancing its ability to learn temporal dependencies within the route.

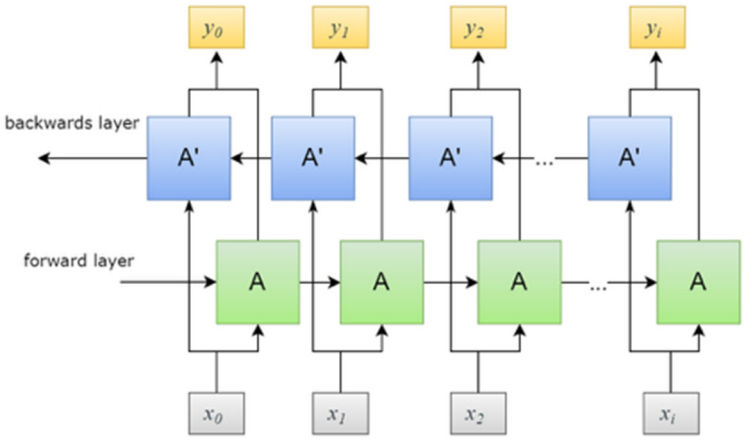


Figure 9. Schematic representation of the BiLSTM network, where the input sequence is processed by two LSTM layers: a forward layer (denoted by A) and a backward layer (denoted by A′).

At each time step i, outputs from the forward and backward LSTM layers are combined via concatenation to form the final representation:

$$p_i = p_i^f + p_i^b\,, \tag{26}$$

where p_i^f and p_i^b denote the forward and backward hidden representations, respectively.

For the proposed model, each step $i \in W_t$, the input feature vector is constructed by concatenating all relevant representations into a unified embedding vector defined as:

$$z_i = \big[Z[s_i],\ Z[e_i],\ E_g[g_i],\ E_q[q_i],\ u_i\big], \tag{27}$$

where

- $Z[s_i]$, $Z[e_i]$ denote *the* rows of the graph-enhanced stop embedding matrix corresponding to the start and end stops of segment i;
- $\mathrm{E}_g[g_i]$ denotes the embedding of the segment identifier;
- $\mathrm{E}_q[q_i]$ denotes the embedding of the segment position within the trip;
- u_i is the continuous feature vector defined in Equation (11).

The resulting historical input sequence for target segment t is defined as:

$$Z_t = \{z_{t-L+1}, \ldots, z_t\}. \tag{28}$$

The sequence Z_t is subsequently processed by a BiLSTM network.

Formally, the BiLSTM processes each step representation z_i and produces a sequence of hidden representations:

$$\mathrm{h}_i = \mathrm{BiLSTM}(z_i), i \in \{t-L+1, \ldots, t\}, \tag{29}$$

where h_i denotes the hidden representation generated at sequence position i. Each hidden state summarizes the encoded contextual information of the corresponding segment after integrating both forward and backward temporal dependencies within the window.

The resulting sequence of hidden states is therefore:

$$H_t = \{\mathrm{h}_{t-\mathrm{L}+1}, \ldots, \mathrm{h}_t\} = \mathrm{BiLSTM}(Z_t). \tag{30}$$

The final hidden state corresponding to the last step is used as the sequence representation:

$$h_t^* = h_t \tag{31}$$

The sequence representation is augmented with global contextual features describing the prediction time as:

$$\mathrm{c}_t = [x_t, E_{r_t}] \tag{32}$$

where E_{r_t} represents the embedding of the route associated with the target segment.

The final prediction vector is obtained by concatenating the sequence representation with the contextual features:

$$h\prime_t = [h_t, c_t] \tag{33}$$

This vector is then passed through a multilayer perceptron (MLP):

$$\hat{\mathrm{y}}_t = f_{MLP}(h\prime_t), \tag{34}$$

where $\hat{\mathrm{y}}_t$ is the predicted travel time for segment t, as in Equation (1).

The model is trained using the mean absolute error (MAE):

$$\mathcal{L} = \frac{1}{N}\sum_{t=1}^{N}|\mathrm{y}_t - \hat{\mathrm{y}}_t|.$$

3.4. Baselines

To evaluate the effectiveness of the proposed graph-enhanced sequential modeling framework (multi-relational graph spatio-temporal bidirectional LSTM, MRG-ST-BiLSTM), its predictive performance was compared against three baseline models: a conventional LSTM, a Hybrid-BiLSTM [30], and a GCN-LSTM model.

The LSTM baseline models the problem as a sequence-to-one regression task using a unidirectional recurrent structure. The input at each time step consists of continuous operational features, complemented by basic categorical embeddings such as segment and route identifiers. The sequence is processed by an LSTM layer, and the final hidden state is passed to a fully connected layer to produce the predicted travel time. It captures temporal dependencies but does not incorporate explicit spatial information about the transport network.

The Hybrid-BiLSTM extends the previous model by using a richer input representation and bidirectional temporal modeling. At each time step, the input additionally combines continuous and temporal features. These representations are processed by a bidirectional LSTM, allowing the model to capture dependencies in both forward and backward directions. The concatenated hidden states are then passed to a prediction layer. While this model improves temporal representation, it does not explicitly model spatial relationships between stops.

The GCN-LSTM incorporates spatial dependencies by introducing a graph-based representation of the transport network. A graph is constructed where nodes correspond to stops and edges represent observed adjacency between consecutive stops. A two-layer GCN is applied to learn spatially enriched stop embeddings. These graph-enhanced embeddings are used in place of the original stop embeddings within the input sequence, which also includes segment, positional, continuous, and temporal features. The resulting sequence is then processed by a unidirectional LSTM, and the final hidden state is used for prediction. This model captures both spatial and temporal dependencies but relies on a single graph structure and forward temporal modeling.

The proposed MRG-ST-BiLSTM model extends the GCN-LSTM framework by introducing both multi-relational spatial modeling and bidirectional temporal encoding. The conceptual differences between the evaluated models are summarized in Table 4.

Table 4. Comparison of the baseline and proposed models.

Model	Input Representation	Spatial Modeling	Temporal Modeling
LSTM	Continuous + basic categorical embeddings	–	LSTM
Hybrid-BiLSTM	Continuous + temporal + richer embeddings	–	BiLSTM
GCN-LSTM	Full features + graph-enhanced stop embeddings	Single graph (adjacency)	LSTM
MRG-ST-BiLSTM	Full features + multi-relational graph embeddings	Multi-relational graph (co-route + adjacency)	BiLSTM

3.5. Model Parameter Configuration

The model was trained using 70/10/20 split training, validation, and testing subsets to avoid temporal data leakage and to simulate real-world prediction conditions. The splits were applied based on timestamps and are summarized in Table 5.

Table 5. Data split distribution.

Split	Date Range	Number of Records
Training set	<23 March 2021	2,474,071
Validation set	23 March 2021–7 April 2021	427,003
Test set	>7 April 2021	754,310

The model parameter settings are presented in Table 6. While core hyperparameters are shared across all models for consistency, the proposed MRG-ST-BiLSTM incorporates additional parameters.

Table 6. Model parameter settings for the baselines and proposed approach.

Parameter	Value	Description
Sequence length	4	Number of historical segments used as input
Batch size	256	Number of samples per training batch
Number of epochs	20	Maximum training epochs
Learning rate	0.001	Initial learning rate for optimizer
Optimizer	Adam	Optimization algorithm
Loss function	MAE (L1 Loss)	Objective function for regression
Early stopping patience	5	Number of epochs without improvement before stopping
Minimum delta	0.01	Minimum improvement threshold for validation MAE
Random seed	42	Ensures reproducibility
LSTM hidden size	64	Hidden dimension of LSTM layers
GCN layers	2	Number of graph convolution layers
GCN hidden dimension	32	Hidden size of GCN
Co-route graph top-k *	8	Number of strongest neighboring stops retained based on shared route similarity

* Only applicable to the proposed model.

3.6. Evaluation Metrics

In this study, the following metrics were used: mean absolute error (MAE), root mean square error (RMSE), mean absolute percentage error (MAPE), symmetric mean absolute percentage error (SMAPE), and the coefficient of determination (R^2) as in Equations (35)–(39). These metrics measure the deviation between the predicted travel times and actual observed values from different perspectives, enabling a comprehensive assessment of model accuracy and robustness.

Mean absolute error (MAE) measures the average magnitude of prediction errors:

$$MAE = \frac{1}{z}\sum_{i=1}^{z}\left|T_i - \acute{T}_i\right|, \tag{35}$$

Root mean square error (RMSE) emphasizes larger prediction errors due to the square operation:

$$RMSE = \sqrt{\left(\frac{1}{z}\sum_{i=1}^{z}\left(T_i - \acute{T}_i\right)^2\right)}. \tag{36}$$

Mean absolute percentage error (MAPE) expresses prediction error as a percentage of the true values:

$$MAPE = \frac{1}{z}\sum_{i=1}^{z}\left|\frac{T_i - \acute{T}_i}{T_i}\right|\cdot 100. \tag{37}$$

Symmetric mean absolute percentage error (SMAPE) reduces the bias caused by very small true values:

$$SMAPE = \frac{1}{z}\sum_{i=1}^{z}\frac{\left|T_i - \acute{T}_i\right|}{\left(\left|T_i\right| + \left|\acute{T}_i\right|\right)/2}\cdot 100. \tag{38}$$

Finally, the coefficient of determination (R^2) evaluates how well the predicted values explain the variance of the true observations:

$$R^2 = 1 - \frac{\sum_{i=1}^{z}(T_i - \acute{T}_i)^2}{\sum_{i=1}^{z}(T_i - \hat{T})^2}, \tag{39}$$

where $T = \{T_1, T_2, \ldots, T_z\}$ denotes the true travel time sequence, $\acute{T} = \{\acute{T}_1, \acute{T}_2, \ldots, \acute{T}_z\}_i$ denotes the predicted travel time sequence, z represents the number of samples in the dataset, and $\hat{T}$ represents the mean value of the true observations.

4. Results

4.1. Main Results

This section presents the experimental evaluation results that demonstrate the effectiveness of the proposed multi-relational graph spatio-temporal modeling framework in improving travel time prediction accuracy. The training dynamics, illustrated in Figure 10, show that all models exhibited a gradual reduction in error during the early training stages, indicating successful learning of temporal patterns. However, the proposed MRG-ST-BiLSTM model consistently achieved the lowest validation MAE and maintained a smaller gap between the training and validation curves compared to the baseline models.

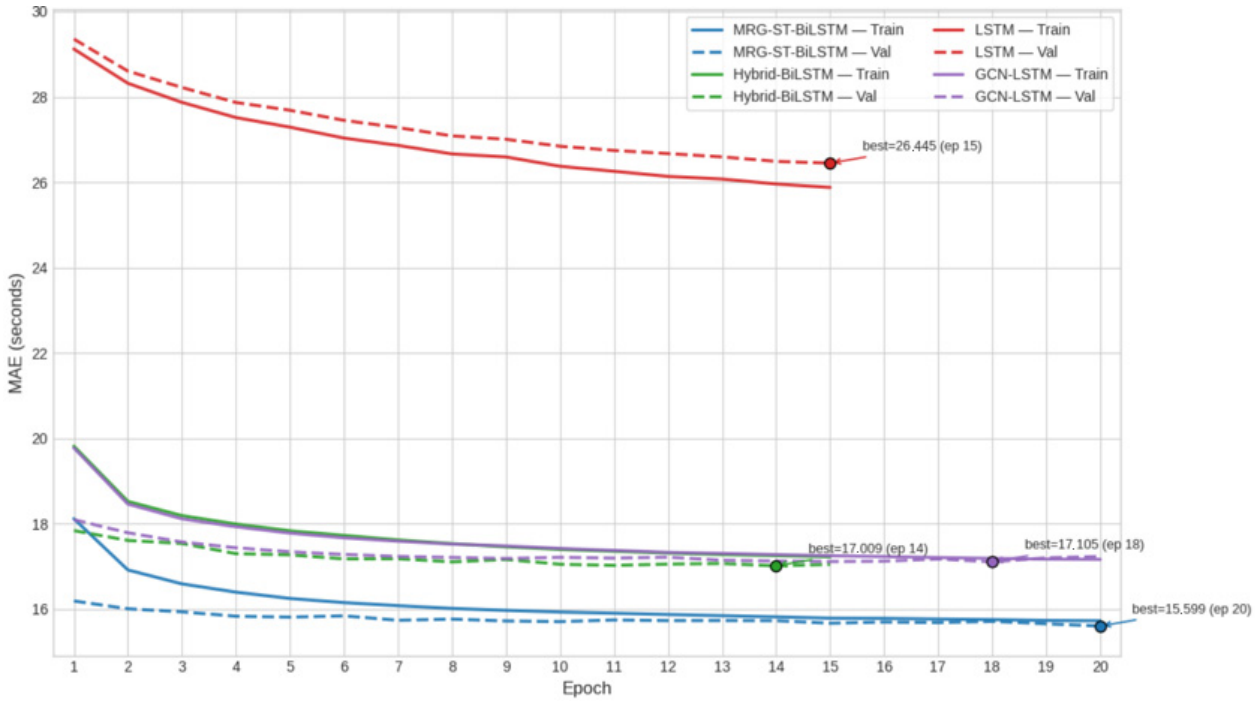


Figure 10. Train and validation loss by epoch.

The quantitative evaluation on the test dataset, as summarized in Table 7, further validated these observations. The conventional LSTM model showed the weakest performance, with an MAE of 26.70 s and an RMSE of 45.61 s, indicating substantial prediction errors. In contrast, the Hybrid-BiLSTM significantly improved performance, reducing MAE to 17.24 s and RMSE to 32.73 s, demonstrating the benefit of incorporating bidirectional temporal dependencies. The GCN-LSTM model also demonstrated competitive performance comparable to the Hybrid-BiLSTM, achieving an MAE of approximately 17.27 s. The proposed MRG-ST-BiLSTM model achieved the best results across all evaluation metrics, with an MAE of 15.82 s and RMSE of 31.38 s. Additionally, it attained the highest coefficient of

determination (R^2 = 0.662), indicating that a larger proportion of variance in travel time is explained by the model.

Table 7. Overall test performance.

Model	MAE (s)	RMSE (s)	MAPE (%)	SMAPE (%)	R^2
LSTM	26.70	45.61	55.97	44.36	0.286
Hybrid-BiLSTM	17.24	32.73	31.84	28.23	0.632
GCN-LSTM	17.27	32.61	31.66	28.19	0.644
MRG-ST-BiLSTM	15.82	31.38	27.54	24.95	0.662

The overall performance comparison across models is further illustrated in Figure 11.

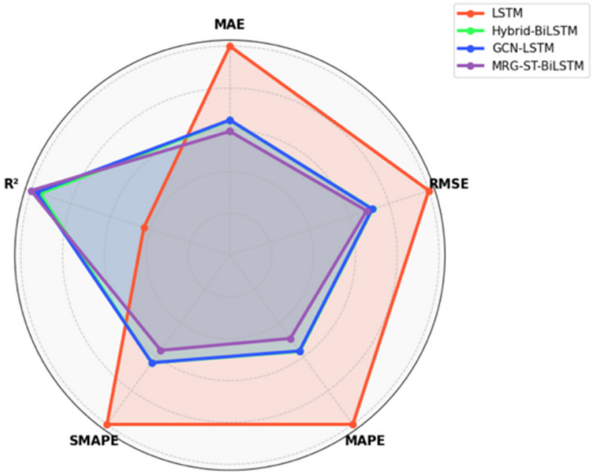


Figure 11. Radar chart illustrating the comparative performance of the models across the evaluation metrics.

A more detailed analysis at the route level, presented in Table 8 and visually illustrated in Figure 12, revealed consistent improvements across different transportation routes.

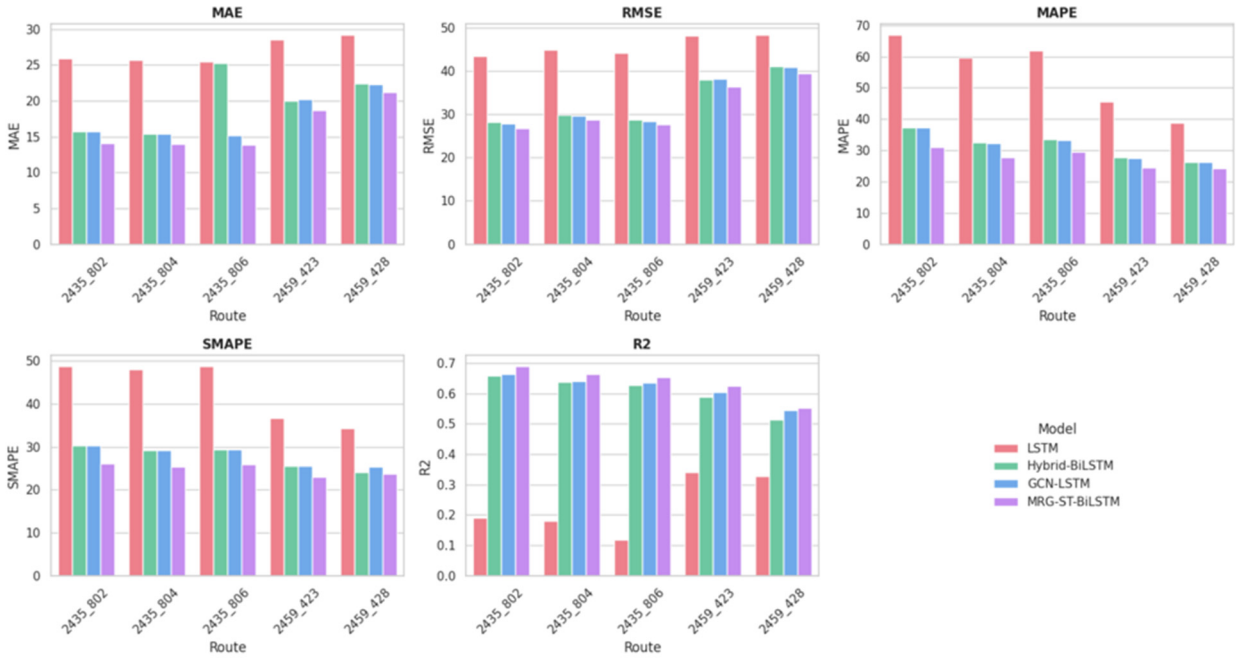


Figure 12. Route-level performance comparison.

Table 8. Per-route comparison of LSTM, Hybrid-BiLSTM, GCN-LSTM, and MRG-ST-BiLSTM models (best in bold).

Route	Model	MAE (s)	RMSE (s)	MAPE (%)	SMAPE (%)	R^2
2435_802	LSTM	25.91	43.43	66.99	48.77	0.191
	Hybrid-BiLSTM	15.74	28.19	37.27	30.24	0.659
	GCN-LSTM	15.67	27.83	37.26	30.25	0.664
	MRG-ST-BiLSTM	**14.12**	**26.83**	**30.99**	**26.06**	**0.691**
2435_804	LSTM	25.73	44.86	59.67	48.05	0.180
	Hybrid-BiLSTM	15.43	29.80	32.54	29.22	0.638
	GCN-LSTM	15.43	29.62	32.38	29.23	0.641
	MRG-ST-BiLSTM	**13.94**	**28.71**	**27.73**	**25.39**	**0.664**
2435_806	LSTM	25.43	44.20	61.92	48.83	0.119
	Hybrid-BiLSTM	25.27	28.70	33.60	29.39	0.629
	GCN-LSTM	15.19	28.35	33.26	29.34	0.636
	MRG-ST-BiLSTM	**13.91**	**27.72**	**29.51**	25.91	**0.653**
2459_423	LSTM	28.55	48.17	45.57	36.67	0.341
	Hybrid-BiLSTM	20.02	38.06	27.73	25.50	0.588
	GCN-LSTM	20.22	38.25	27.50	25.48	0.604
	MRG-ST-BiLSTM	**18.66**	**36.29**	**24.39**	**23.03**	**0.626**
2459_428	LSTM	29.22	48.38	38.70	34.28	0.327
	Hybrid-BiLSTM	22.40	41.14	26.22	24.13	0.514
	GCN-LSTM	22.34	40.86	26.27	25.28	0.546
	MRG-ST-BiLSTM	**21.24**	**39.45**	**24.19**	**23.79**	**0.553**

To assess the computational overhead of the evaluated models, we compared them in terms of parameter count, model size, training time, inference latency, and memory consumption. As shown in Table 9, the LSTM was the most lightweight and fastest at inference (0.043 ms/sample) but had the weakest predictive performance. The Hybrid-BiLSTM significantly increased complexity, with over four times more parameters and the highest inference latency (0.067 ms). Furthermore, the GCN-LSTM provided a more balanced trade-off, incorporating spatial information with moderate computational cost and near-baseline inference speed (0.047 ms). The proposed MRG-ST-BiLSTM had the highest complexity in terms of parameters and memory, however, it maintained efficient inference (0.046 ms) and achieved the lowest total training time (3635.3 s).

Table 9. The computational overhead of all compared models.

Model	Parameters	Average Epoch Time (s)	Total Training Time (s)	Inference (ms/Sample)	Model Size (MB)	CPU Memory (MB)
LSTM	25,241	166.6	4470.8	0.043	0.096	1967
Hybrid-BiLSTM	103,433	318.9	3872.2	0.067	0.395	3775
GCN-LSTM	68,083	225.3	4506.7	0.047	0.260	3606
MRG-ST-BiLSTM	104,457	242.3	3635.3	0.046	0.398	3921

Overall, despite the increased model complexity, the proposed model preserves low inference latency while delivering superior predictive performance.

4.2. Error Analysis Across Peak Periods and Irregular Traffic Conditions

To better understand model behavior beyond aggregate test metrics, we conducted an additional error analysis across peak and off-peak time periods and irregular traffic conditions, using a temporally separated test set not seen during training. Following the opera-

tional characteristics of urban bus services, weekday records were divided into peak hours (07:00–09:00, 15:00–17:00) and off-peak hours, while weekends were analyzed separately.

Table 10 presents the comparative error metrics across different settings. It shows that all models exhibited higher errors during peak periods than during off-peak periods. However, the proposed MRG-ST-BiLSTM consistently achieved the lowest MAE across all temporal subsets. Specifically, its MAE was 17.95 s during peak hours, outperforming LSTM (30.03 s), Hybrid-BiLSTM (24.00 s), and GCN-LSTM (25.21 s). Under off-peak conditions, it achieved an MAE of 15.09 s, compared to 25.46 s for LSTM, 21.11 s for Hybrid-BiLSTM, and 19.45 s for GCN-LSTM. Similarly, during weekends, the proposed model obtained an MAE of 16.04 s, while the corresponding errors for LSTM, Hybrid-BiLSTM, and GCN-LSTM were 25.96, 22.12, and 20.60 s, respectively.

Table 10. Comparison of model performance across temporal periods (peak, off-peak, and weekend) in terms of MAE, RMSE, and MAPE.

Model	Period	MAE (s)	RMSE (s)	MAPE (%)
LSTM	Peak	30.03	52.66	51.81
	Off-peak	25.46	42.58	55.72
	Weekend	25.96	44.74	61.90
Hybrid-BiLSTM	Peak	24.00	42.83	24.83
	Off-peak	21.11	35.45	27.18
	Weekend	22.12	38.39	32.85
GCN-LSTM	Peak	25.21	43.04	28.79
	Off-peak	19.45	33.32	31.03
	Weekend	20.60	37.30	36.63
MRG-ST-BiLSTM	Peak	17.95	36.58	24.87
	Off-peak	15.09	29.39	27.08
	Weekend	16.04	31.75	32.47

The results are further illustrated in Figure 13.

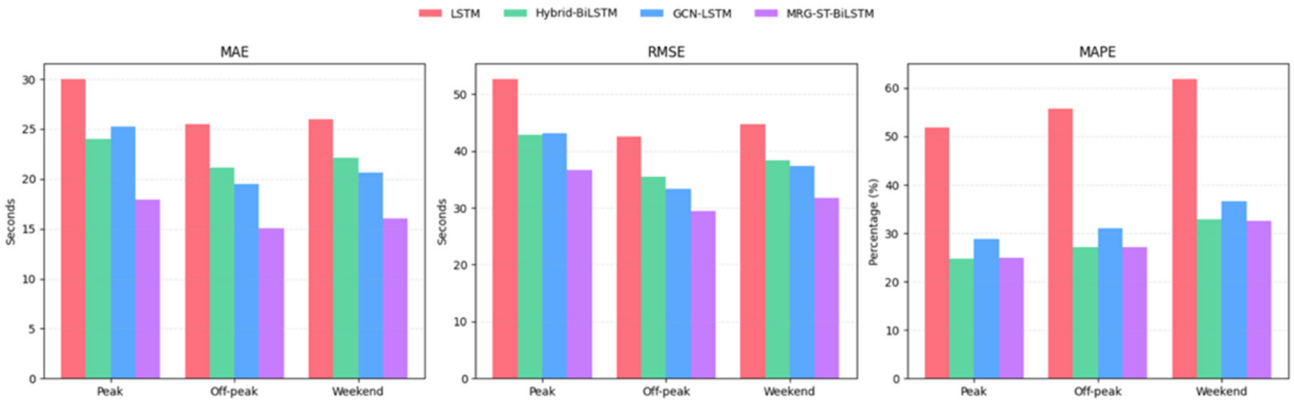


Figure 13. Comparison of model performance peak, off-peak, and weekend in terms of MAE, RMSE, and MAPE metrics.

In addition, to examine model robustness under non-periodic conditions, we identified an irregular subset using a residual-based criterion, where samples with absolute error greater than the period-specific threshold of $\mu + 2\sigma$ were treated as high-error cases. Table 11 summarizes the error characteristics for both regular and irregular subsets.

Table 11. Prediction errors under regular and irregular traffic conditions.

Model	Period	Irregular (%)	Regular MAE (s)	Irregular MAE (s)
LSTM	Peak	4.13	23.36	184.78
	Off-peak	4.04	20.35	146.51
	Weekend	3.53	20.99	161.73
Hybrid-BiLSTM	Peak	3.30	19.76	149.26
	Off-peak	3.10	18.05	117.73
	Weekend	2.60	19.09	136.69
GCN-LSTM	Peak	3.26	21.07	148.05
	Off-peak	3.09	16.35	116.65
	Weekend	2.47	17.49	143.52
MRG-ST-BiLSTM	Peak	3.29	13.75	141.30
	Off-peak	3.09	12.03	111.07
	Weekend	2.64	13.06	126.12

Across all models and temporal subsets, irregular samples accounted for approximately 2.5–4% of observations, indicating that such events are relatively infrequent, but they lead to a substantial increase in prediction error. Specifically, MAE under irregular conditions was approximately 6–10 times higher than under regular conditions across all models, indicating that prediction errors are highly concentrated in a limited number of high-variance events.

Moreover, the proposed MRG-ST-BiLSTM consistently achieved the lowest MAE in both regular and irregular subsets across all temporal periods. For example, under peak conditions, it reduced the irregular MAE from 184.78 s (LSTM) to 141.30 s while also maintaining the lowest error under regular conditions. This trend was consistently observed for the off-peak and weekend scenarios. Overall, these results demonstrate that the proposed model not only improves the prediction accuracy under typical traffic conditions but also exhibits greater robustness to anomalous and non-periodic events.

4.3. The Ablation Study

An ablation study was conducted to analyze the effect of the graph size and historical window size on model performance. Table 12 reports the results of the ablation study on the graph size parameter (top-k), evaluating both predictive performance and computational efficiency. The k = 2 configuration produced the weakest results, with the highest errors (MAE = 20.01 s, RMSE = 35.08 s, MAPE = 34.55%) and the lowest explanatory power (R^2 = 0.628). This confirms that an overly sparse graph (3727 edges) is insufficient to model spatial dependencies between stops. Increasing the graph size to k = 8 significantly improved the performance across all while also maintaining relatively efficient computation (average epoch time = 242.3 s, inference = 0.046 ms/sample). This configuration achieved the best trade-off between accuracy and efficiency. Further increasing the graph size to k = 20 resulted in a much denser graph (29,119 edges) and substantially higher computational cost (epoch time = 365.5 s; total training time = 7311.2 s) while yielding no meaningful improvement in accuracy. The results suggest that excessive connectivity may introduce noise and reduce generalization.

The historical window is defined as the number of previous time steps (sequence length) used as input for prediction. To evaluate its influence, the sequence length was varied from 1 to 5, and the performance of all models was assessed using MAE, RMSE, and SMAPE, as presented in Figure 14.

Table 12. Ablation study on the graph size parameter.

Top-k	MAE (s)	RMSE (s)	MAPE (%)	R^2	Edges (A Graph)	Avrg Epoch Time (s)	Inference (ms/Sample)	Parameters	Total Training Time (s)	Model Size (MB)	CPU Memory (MB)
2	20.01	35.08	34.55	0.628	3727	331.2	0.062	104,457	3643.4	0.398	3917
8	15.82	31.38	27.54	0.662	12,521	242.3	0.046		3635.3		3921
20	15.84	31.46	27.63	0.660	29,119	365.5	0.072		7311.2		3926

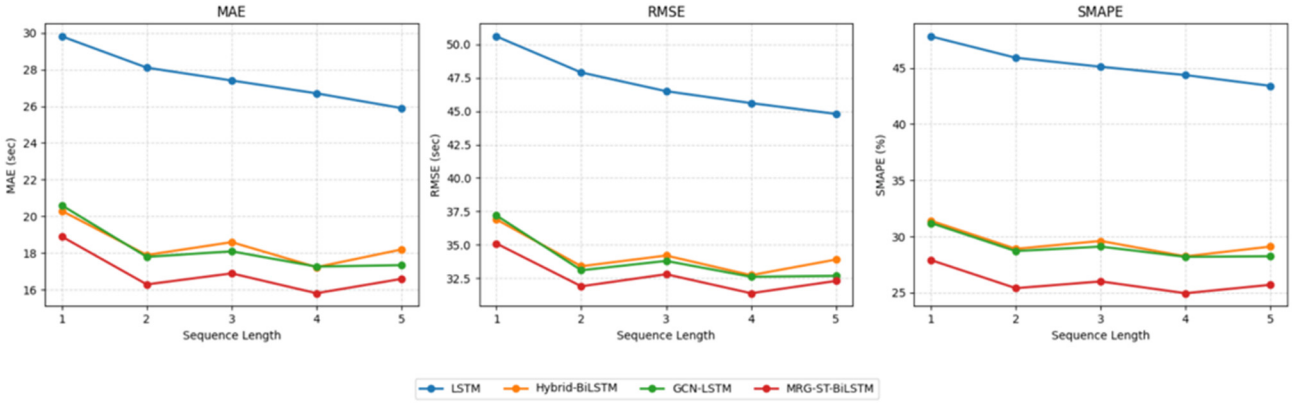


Figure 14. Performance comparison of models under varying historical window sizes.

Increasing the historical window from 1 to 2 led to a substantial reduction in metrics. Further increasing the sequence length resulted in more moderate improvements, with most models achieving their best performance at a window size of 4. Beyond this point, extending the sequence length to 5 did not yield additional gains and in some cases slightly degraded the performance, suggesting the introduction of redundant or less informative temporal signals. Across all configurations, the proposed model consistently achieved the best performance compared to the baselines.

5. Discussion and Conclusions

In contrast to data-intensive approaches, the proposed model is formulated as a spatiotemporal learning problem defined over a structured transportation network, where temporal dynamics are captured through sequential modeling and spatial interactions are represented via a multi-relational graph. From the standpoint of the research questions, the results provide clear answers. For RQ1, the model achieved strong predictive accuracy using only standard operational data, indicating that such data contain sufficient information to approximate short-term system dynamics when properly modeled. Furthermore, the reliance on the universal GTFS standard ensures high portability, allowing the framework to be deployed in any urban transit system with standard operational feeds. For RQ2, the improvement observed when introducing multi-relational graph dependencies confirmed that spatial interactions between route segments and their explicit mathematical representation leads to measurable gains in performance. Critically, this graph-based formulation offers topological resilience; unlike fixed-grid models, our framework allows for 'plug-and-play' updates to the adjacency matrix when routes are modified or segments are temporarily closed, without requiring full retraining of the model.

The findings of this research offer several practical insights for transit agencies and urban transport planners. First, it demonstrates that effective travel time prediction does not necessarily require investments in costly sensor networks. For many agencies, the operational data already collected through automatic vehicle location and scheduling systems

are sufficient to build high-performing predictive models. This lowers the barrier to entry for implementing intelligent transportation systems in resource-constrained environments. Second, the graph-enhanced approach provides a degree of structural interpretability that is often absent in 'black-box' deep learning models. By explicitly modeling the network topology, predictions can be traced back to spatial relationships between route segments. For instance, if a delay is predicted on a particular segment, operators can examine the state of its neighboring segments to understand the potential propagation path. This transparency is crucial for building trust in automated forecasting systems and for supporting human-in-the-loop decision-making in fleet management.

While this study makes significant contributions, it is not without limitations. Although the reliance exclusively on operational data was a deliberate design choice to ensure data efficiency, it means that the model does not account for exogenous factors that can significantly impact travel time, such as special events, traffic incidents, or real-time passenger demand. While the proposed framework allows for rapid topological updates via GTFS, its current version still relies on historical patterns to weight these connections. Therefore, the model's performance may degrade during periods of extreme disruption that are not well-represented in the historical operational data. Furthermore, while the multi-relational graph captures fixed and sequential similarities, it does not yet incorporate dynamic edge weights that could reflect real-time link capacity variations.

Building on the findings and limitations of this study, several avenues for future research emerge. A primary direction is the expansion of the model to a large-scale, city-wide network. Another promising direction is to develop a flexible modular framework that allows for the integration of stochastic external variables and different data sources. While a key contribution of this work is its data efficiency, a natural extension is to develop a flexible framework that can optionally incorporate external variables when they are available. This would allow the model to maintain its core data efficiency while potentially improving accuracy during non-recurrent congestion events. Finally, integrating an attention-based mechanism to dynamically adjust graph weights in response to real-time anomalies remains a key objective for future work.

In conclusion, this study contributes to the field of transportation modeling by proposing a topology-aware spatiotemporal framework that combines sequential learning with graph-based representation of network structure. The results demonstrate that accurate and scalable prediction can be achieved through principled mathematical modeling, without reliance on extensive external data sources. This work provides a step toward bridging the gap between data-driven methods and formal transportation system modeling.

Author Contributions: Conceptualization, A.M. and A.A.; methodology, A.M. and A.A.; validation, A.M.; formal analysis, A.M.; data curation, D.R.; writing—original draft preparation, A.A. and A.M.; writing—review and editing, D.Y.; visualization, A.M. and A.A.; supervision, D.Y.; project administration, D.Y.; funding acquisition, D.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research has been funded by the Committee of Science of the Ministry of Science and Higher Education of the Republic of Kazakhstan (Grant No. BR24992852 "Intelligent models and methods of Smart City digital ecosystem for sustainable development and the citizens' quality of life improvement").

Data Availability Statement: These data were derived from the following resources available in the public domain: https://sydney.primo.exlibrisgroup.com/permalink/61USYD_INST/1tvk8cv/alma991032393325705106 (accessed on 30 March 2026).

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ANN	Artificial Neural Network
APC	Article Processing Charge
ARIMA	Auto Regressive Integrated Moving Average
BiLSTM	Bidirectional Long Short-Term Memory
CNN	Convolutional Neural Network
ESC	Electronic Smart Card
GCN	Graph Convolutional Network
GPS	Global Positioning System
GTFS	General Transit Feed Specification
ITS	Intelligent Transportation Systems
KF	Kalman Filter
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
MAPE	Mean Absolute Percentage Error
MRG-ST-BiLSTM	Multi-Relational Graph Spatiotemporal Bidirectional LSTM
RMSE	Root Mean Square Error
SMAPE	Symmetric Mean Absolute Percentage Error
SVR	Support Vector Regression

References

1. Ge, L.; Sarhani, M.; Voß, S.; Xie, L. Review of Transit Data Sources: Potentials, Challenges and Complementarity. *Sustainability* **2021**, *13*, 11450. [CrossRef]
2. Yang, X.; Yang, H.; Mao, Y.; Wang, Q.; Yin, S. Multi-Task Dynamic Spatio-Temporal Graph Attention Network: A Variable Taxi Time Prediction Model for Airport Surface Operation. *Aerospace* **2024**, *11*, 371. [CrossRef]
3. Ye, J.; Xue, S.; Jiang, A. Attention-based spatio-temporal graph convolutional network considering external factors for multi-step traffic flow prediction. *Digit. Commun. Netw.* **2022**, *8*, 343–350. [CrossRef]
4. Rong, Y.; Yao, J.; Liu, J.; Fang, Y.; Luo, W.; Liu, H.; Ma, J.; Dan, Z.; Lin, J.; Wu, Z.; et al. Gbtte: Graph attention network based bus travel time estimation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*; Association for Computing Machinery: Birmingham, UK, 2023; pp. 4794–4800. [CrossRef]
5. Li, D.; Lasenby, J. Spatiotemporal attention-based graph convolution network for segment-level traffic prediction. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 8337–8345. [CrossRef]
6. Chen, L.; Wu, Q. Spatiotemporal-decoupled interactive learning for traffic flow prediction. *Sci. Rep.* **2026**, *16*, 9050. [CrossRef] [PubMed]
7. Maiti, S.; Pal, A.; Pal, A.; Chattopadhyay, T.; Mukherjee, A. Historical Data Based Real Time Prediction of Vehicle Arrival Time. In *Proceedings of the 17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*; IEEE: Qingdao, China, 2014; pp. 1837–1842. [CrossRef]
8. Wepulanon, P.; Sumalee, A.; Lam, W.H.K. A Real-Time Bus Arrival Time Information System Using Crowdsourced Smartphone Data: A Novel Framework and Simulation Experiments. *Transp. B Transp. Dyn.* **2018**, *6*, 34–53. [CrossRef]
9. Taparia, A.; Brady, M. Bus Journey and Arrival Time Prediction Based on Archived AVL/GPS Data Using Machine Learning. In *Proceedings of the 2021 7th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*; IEEE: Heraklion, Greece, 2021; pp. 1–6. [CrossRef]
10. Zhou, W.X.; Xu, J.M.; Liu, Z.D. Forecasting of Travel Time for Public Transport Vehicles Based on the Kalman Filtering Algorithm. *Commun. Stand.* **2007**, *2007*, 174–177.
11. Sinn, M.; Yoon, J.W.; Calabrese, F.; Bouillet, E. Predicting Arrival Times of Buses Using Real-Time GPS Measurements. In *Proceedings of the 2012 15th International IEEE Conference on Intelligent Transportation Systems*; IEEE: Anchorage, AK, USA, 2012; pp. 1227–1232. [CrossRef]
12. Fan, W.; Gurmu, Z. Dynamic Travel Time Prediction Models for Buses Using Only GPS Data. *Int. J. Transp. Sci. Technol.* **2015**, *4*, 353–366. [CrossRef]
13. Achar, A.; Bharathi, D.; Kumar, B.A.; Vanajakshi, L. Bus Arrival Time Prediction: A Spatial Kalman Filter Approach. *IEEE Trans. Intell. Transp. Syst.* **2020**, *21*, 1298–1307. [CrossRef]

14. Kumar, B.A.; Vanajakshi, L.; Subramanian, S.C. Bus Travel Time Prediction Using a Time-Space Discretization Approach. *Transp. Res. Part C Emerg. Technol.* **2017**, *79*, 308–332. [CrossRef]
15. Suwardo, W.; Napiah, M.; Kamaruddin, I. ARIMA Models for Bus Travel Time Prediction. *J. Inst. Eng. Malays.* **2010**, *71*, 49–58.
16. Mete, S.; Çelik, E.; Gül, M. Predicting the Time of Bus Arrival for Public Transportation by Time Series Models. *J. Transp. Logist.* **2022**, *7*, 441–455. [CrossRef]
17. Xiang, G.; Deyong, G.; Yuxuan, X. GA-ARIMA Model-Based Analysis of Arrival Time at Bus Stop. *IOP Conf. Ser. Earth Environ. Sci.* **2020**, *587*, 012050. [CrossRef]
18. Bajčetić, S.; Živanović, P.; Tica, S.; Milovanović, B.; Nađ, A. Factors Influencing Driving Time in Public Transport—A Multiple Regression Analysis. *Promet* **2023**, *35*, 37–54. [CrossRef]
19. Ashwini, B.P.; Sumathi, R.; Sudhira, H.S. Bus Travel Time Prediction: A Comparative Study of Linear and Non-Linear Machine Learning Models. *J. Phys. Conf. Ser.* **2022**, *2161*, 012053. [CrossRef]
20. Yu, B.; Lam, W.H.K.; Tam, M.L. Bus Arrival Time Prediction at Bus Stop with Multiple Routes. *Transp. Res. Part C Emerg. Technol.* **2011**, *19*, 1157–1170. [CrossRef]
21. Kviesis, A.; Zacepins, A.; Komasilovs, V.; Munizaga, M. Bus Arrival Time Prediction with Limited Data Set Using Regression Models. In *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems*; SCITEPRESS—Science and Technology Publications: Funchal, Portugal, 2018; pp. 643–647. [CrossRef]
22. Khadhir, A.; Anil Kumar, B.; Vanajakshi, L.D. Analysis of Global Positioning System Based Bus Travel Time Data and Its Use for Advanced Public Transportation System Applications. *J. Intell. Transp. Syst.* **2021**, *25*, 58–76. [CrossRef]
23. Osman, O.; Rakha, H.; Mittal, A. Application of Long Short Term Memory Networks for Long- and Short-Term Bus Travel Time Prediction. *Preprints* **2021**. [CrossRef]
24. He, P.; Jiang, G.; Lam, S.-K.; Tang, D. Travel-Time Prediction of Bus Journey with Multiple Bus Trips. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 4192–4205. [CrossRef]
25. Alam, O.; Kush, A.; Emami, A.; Pouladzadeh, P. Predicting Irregularities in Arrival Times for Transit Buses with Recurrent Neural Networks Using GPS Coordinates and Weather Data. *J. Ambient Intell. Humaniz. Comput.* **2021**, *12*, 7813–7826. [CrossRef]
26. Hou, Y.; Edara, P. Network Scale Travel Time Prediction Using Deep Learning. *Transp. Res. Rec. J. Transp. Res. Board* **2018**, *2672*, 115–123. [CrossRef]
27. Shanthi, N.; Suganthe, R.C.; Manimaran, V.; Manoj Kumar, P.; Kavin Bharathi, A.; Mythili, E. GPS-Based Arrival Time Prediction for Bus Schedules: Enhancing Commuter Experience. In *Proceedings of the 2023 12th International Conference on Advanced Computing (ICoAC)*; IEEE: Chennai, India, 2023; pp. 1–6. [CrossRef]
28. Han, Q.; Liu, K.; Zeng, L.; He, G.; Ye, L.; Li, F. A Bus Arrival Time Prediction Method Based on Position Calibration and LSTM. *IEEE Access* **2020**, *8*, 42372–42383. [CrossRef]
29. Yuan, Y.; Shao, C.; Cao, Z.; He, Z.; Zhu, C.; Wang, Y.; Jang, V. Bus Dynamic Travel Time Prediction: Using a Deep Feature Extraction Framework Based on RNN and DNN. *Electronics* **2020**, *9*, 1876. [CrossRef]
30. Mansurova, A.; Rakhimzhanov, D.; Mansurova, A. Bus Travel Time Prediction Under Data Constraints: A Lightweight Deep Learning Approach. In *2025 International Conference on Computational Intelligence and Knowledge Economy (ICCIKE)*; IEEE: Dubai, United Arab Emirates, 2025; pp. 444–449. [CrossRef]
31. Gal, A.; Mandelbaum, A.; Schnitzler, F.; Senderovich, A.; Weidlich, M. Traveling Time Prediction in Scheduled Transportation with Journey Segments. *Inf. Syst.* **2017**, *64*, 266–280. [CrossRef]
32. Wu, J.; Wu, Q.; Shen, J.; Cai, C. Towards Attention-Based Convolutional Long Short-Term Memory for Travel Time Prediction of Bus Journeys. *Sensors* **2020**, *20*, 3354. [CrossRef] [PubMed]
33. Ratneswaran, S.; Thayasivam, U. An Improved Bus Travel Time Prediction Using Multi-Model Ensemble Approach for Heterogeneous Traffic Conditions. In *Proceedings of the 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*; IEEE: Bilbao, Spain, 2023; pp. 2410–2415. [CrossRef]
34. Petersen, N.C.; Rodrigues, F.; Pereira, F.C. Multi-Output Bus Travel Time Prediction with Convolutional LSTM Neural Network. *Expert Syst. Appl.* **2019**, *120*, 426–435. [CrossRef]
35. Lee, G.; Choo, S.; Choi, S.; Lee, H. Does the Inclusion of Spatio-Temporal Features Improve Bus Travel Time Predictions? A Deep Learning-Based Modelling Approach. *Sustainability* **2022**, *14*, 7431. [CrossRef]
36. Liu, H.; Xu, H.; Yan, Y.; Cai, Z.; Sun, T.; Li, W. Bus Arrival Time Prediction Based on LSTM and Spatial-Temporal Feature Vector. *IEEE Access* **2020**, *8*, 11917–11929. [CrossRef]
37. Jiang, R.; Hu, D.; Chien, S.I.-J.; Sun, Q.; Wu, X. Predicting Bus Travel Time with Hybrid Incomplete Data—A Deep Learning Approach. *Promet* **2022**, *34*, 673–685. [CrossRef]
38. Li, C.; Lin, S.; Zhang, H.; Zhao, H.; Liu, L.; Jia, N. A sequence and network embedding method for bus arrival time prediction using GPS trajectory data only. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 5024–5038. [CrossRef]
39. Huang, Y.; Song, X.; Zhang, S.; Li, L.; Yu, J.J. GT-TTE: Modeling trajectories as graphs for travel time estimation. *IEEE Internet Things J.* **2024**, *11*, 30965–30977. [CrossRef]

40. Lei, J.; Chen, Y.; Han, Q.; Zeng, L.; He, G. Effective Bus Travel Time Prediction System of Multiple Routes: Introducing PMLNet Based on MDARNN. *Appl. Sci.* **2025**, *15*, 8104. [CrossRef]
41. Qiu, T.; Lam, C.T.; Liu, B.; Ng, B.K.; Yuan, X.; Im, S.K. FEN-MRMGCN: A frontend-enhanced network based on multi-relational modeling GCN for bus arrival time prediction. *IEEE Access* **2025**, *13*, 5296–5307. [CrossRef]
42. Xian, T.; Chin, T.K.; Marks, B.; Nelson, J.D.; Moylan, E. Bus arrival and departure time updates in the Greater Sydney Area. *Sci. Data* **2024**, *11*, 1034. [CrossRef] [PubMed]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.