International Journal of Transportation Science and Technology xxx (xxxx) xxx

Contents lists available at ScienceDirect

International Journal of Transportation Science and Technology

journal homepage: www.elsevier.com/locate/ijtst

A GTFS data acquisition and processing framework and its application to train delay prediction

Jianqing Wu [a], Bo Du [b,c], Zengyang Gong [d], Qiang Wu [e], Jun Shen [a,*], Luping Zhou [f], Chen Cai [g]

[a] School of Computing and Information Technology, University of Wollongong, NSW 2522, Australia
[b] SMART Infrastructure Facility, University of Wollongong, NSW 2522, Australia
[c] School of Civil, Mining and Environmental, University of Wollongong, NSW 2522, Australia
[d] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong 999077, China
[e] Institute of Fundamental and Frontier Sciences, University of Electronic Science and Technology of China, Chengdu 611731, China
[f] School of Electrical and Information Engineering, University of Sydney, NSW 2006, Australia
[g] CSIRO's Data61, NSW 2015, Australia

ARTICLE INFO

Article history:
Received 3 September 2021
Received in revised form 22 December 2021
Accepted 13 January 2022
Available online xxxx

Keywords:
General transit feed specification
Delay prediction
Train delay
Long short-term memory
Data fusion

ABSTRACT

With advanced artificial intelligence and deep learning techniques, a growing number of data sources are playing more and more critical roles in planning and operating transportation services. The General Transit Feed Specification (GTFS), with standard open-source data in both static and real-time formats, is being widely used in public transport planning and operation management. However, compared to other extensively studied data sources such as smart card data and GPS trajectory data, the GTFS data lacks proper investigation yet. Utilization of the GTFS data is challenging for both transport planners and researchers due to its difficulty and complexity of understanding, processing, and leveraging the raw data. In this paper, a GTFS data acquisition and processing framework is proposed to offer an efficient and effective benchmark tool for converting and fusing the GTFS data to a ready-to-use format. To validate and test the proposed framework, a multivariate multi-step Long Short-Term Memory is developed to predict train delay with minor anomaly in Sydney as a case study. The contribution of this new framework will render great potential for broader applications and deeper research.

1. Introduction

Over the last decade, multimodal transport has become an increasingly popular and influential topic in smart cities. As one of the essential components of Intelligent Transportation Systems (ITS), Advanced Traveler Information System provides real-time traffic information to travelers. Evidence indicated that good quality real-time information leads to increased rider satisfaction and ridership (Barbeau, 2018; Brakewood et al., 2015; Tang and Thakuriah, 2012). For example, the multimodal transport ecosystem of New South Wales (NSW) in Australia provides General Transit Feed Specification (GTFS) static and real-time data to transport planners and operators (Transport for NSW). The GTFS data can offer real observations as multivariate time-series data. Based on the real-time traffic information, travelers can plan their trips easily, and likewise,

Peer review under responsibility of Tongji University and Tongji University Press.
* Corresponding author.
E-mail address: jshen@uow.edu.au (J. Shen).

https://doi.org/10.1016/j.ijtst.2022.01.005
2046-0430/

transport operators can schedule and coordinate transport services to offer better services. Moreover, GTFS has almost no restrictions for registered users and no privacy issues. In the era of big data, it provides a continuous stream of real-time data for researchers to explore the real time information of a city transport network in large scale from both spatial and temporal perspectives.

Nowadays, with the broad deployment of smart city infrastructures, a large number of Internet of Things devices also generate massive data. To facilitate data-driven research, various data sources such as automatic fare collection (AFC) data, automatic vehicle location (AVL), and automatic passenger counting (APC) are widely used. For instance, the AFC data collected from smart cards can be employed to derive an accurate origin–destination (OD) matrix to represent travel demand. The AVL data shows a detailed representation of supply on corresponding trips with planned time and vehicle location. The APC data provides accurate counts of ridership on vehicles. However, limited access to those datasets becomes a major obstacle for widely usage of the data.

Different from aforementioned data sources, GTFS data received rare investigation in literature studies. Moreover, due to the large amount of GTFS data, it is very difficult and complex to process the data, and fuse the static and real-time data for further usage. Therefore, as the first step, how to acquire and process the large-amount and complicated GTFS raw data to support further development of data-driven models and algorithms remains an essential but critical issue. To the best of our knowledge, limited existing research had ever made maneuvers to tackle this foundation issue. Therefore, in this study, we aim to reinvigorate the GTFS data by proposing a GTFS data acquisition and processing (DAP) framework as a foundation to support a diversity of data-driven studies across a broad research scope. The main contributions of this paper is threefold:

(1) A GTFS DAP framework is designed to acquire, process and fuse GTFS static and real-time data from cloud-based live feeds;
(2) Based on the GTFS DAP framework, a data cleaning and aggregation tool is developed to generate benchmark datasets for real-time delay prediction and long-term delay prediction, respectively;
(3) The proposed GTFS DAP framework and generated benchmark dataset are utilized for time series prediction using multivariate multistep Long Short-Term Memory (MM-LSTM). Numerical experiments are conducted based on a case study of Sydney trains with promising results, which validate effectiveness and benefits of the proposed framework.

The remainder of the paper is organized as follows. Related work is introduced in Section 2. Section 3 describes a GTFS DAP framework, including framework design and data cleaning. Section 4 presents the experiment results using a train line in Sydney as a case study. Section 5 concludes the paper with potential future work.

2. Literature review

Cities can develop and implement intelligent analysis systems to monitor the performance of public transit by integrating, analyzing and modeling real-world data (Zeng et al., 2014). In recent years, it is a common practice that deep learning models, as a classifier or predictor, can enhance the accuracy in a significant number of ITS applications (Wang et al., 2019). Relying on the completeness of datasets, deep learning models can derive patterns and models from a large amount of data (Zhu et al., 2019). Data fusion was widely used to form labelled training samples, for example, through combining mobility data with survey data, or subway smart card data and taxi GPS data with mobile phone signaling data (Huang et al., 2018; Kusakabe and Asakura, 2014; Zhao et al., 2020).

GTFS data processing: In general, data cleaning activities consist of error detection and error repair (Ilyas and Chu, 2019). Outlier detection techniques for error detection of time series data can be divided into two main types to deal with two kinds of issues from a computational perspective, including outliers crossing over a time-series database and outliers within a single time series (Gupta et al., 2013). Given an available time series, it also contains two cases, namely point outliers (particular elements) and subsequence outliers. GTFS real-time Trip Update APIs provide daily delay information, which includes abnormal delays. Furthermore, the abnormal event is usually unforeseen or unpredictable. Such events usually have features of suddenness and uncertainty, therefore, proper methodologies are needed to find point outliers in a time series.

For multivariate time series data, prediction models can directly identify outliers for all constituent time series (Gupta et al., 2013), such as multilayer perceptron (MLP) (Hill and Minsker, 2010), mixture transition distribution (MTD) (Le et al., 1996), and autoregressive integrated moving average (ARIMA) (Tsay et al., 2000). However, Bayesian structural time series (BSTS) can flexibly adapt to various assumptions on the latent states of the observed data, including local trends and seasonality (Brodersen et al., 2015). Also, a Bayesian method was used to model the temporal evolution of observation data and utilizes a regression to avoid overfitting. BSTS was a statistical technique that can be applied for feature selection, time series forecasting, nowcasting, and inferring causal impact (Brodersen et al., 2015; Peters et al., 2017; Scott and Varian, 2014, 2015). BSTS can also be extended to deal with inference and prediction for multiple correlated time series (Jammalamadaka et al., 2018). Training a Bayesian model required building a posterior that factors over model parameters, such as Markov Chain Monte Carlo (MCMC) and variational inference (VI) (Johnson and Willsky, 2014; Scott and Varian, 2014). Furthermore, data imputation is another common task in data analysis, which fixes missing or empty values. For a GTFS dataset, these

values invalidate the record, and they can be the exact string *NaN*, or the number 0. No matter how these values appear in the dataset, understanding what to expect, and checking consistency whether the data matches that expectation, will reduce potential issues in using the information later on (McCallum, 2012).

GTFS-based studies: Smart card data were collected by the AFC system that implies valuable knowledge about human travel patterns (Mahrsi et al., 2017; Zheng et al., 2014). The availability of smart card data was of vital importance for answering different research questions in ITS, such as the estimations of the OD distribution, the demand forecasting, the citywide crowd flows forecast, and the bus bunching identification and prediction (Du, 2019; Du and Dublanche, 2018; Gong et al., 2020; Pelletier et al., 2011; Zheng et al., 2014). Specifically, travel demand obtained from the archived smart card transaction records with a data provision based on GTFS schedules can generate highly detailed representations for bus network microsimulation (Gaudette et al., 2016). Advanced deep learning algorithms learn from the fusion of GTFS static and real-time data to solve time series forecasting tasks, such as train delay prediction and travel time foresting of bus journeys (Wu et al., 2020; Wu et al., 2019).

Compared to AVL data, AFC data with GTFS data can accelerate the search process for generating O-D matrices (Alsger et al., 2016; Nassir et al., 2011). Smart card data from APC systems and GTFS data can be fused to visualize public transit use (Giraud et al., 2016). Since GTFS data is more accessible than AVL data, a robust trip chaining method can use probability distributions to infer the most likely trajectory of individual transit passengers. Furthermore, GTFS data can be utilized to provide a list of candidate stops with scheduled time to find the closest stop to the current tag location from AFC data (Kumar et al., 2018). Smart card data can be utilized with GTFS static and other data sources to reduce the use of the expensive and time-consuming Household Travel Surveys (HTS) for inferring passengers' trip purpose (Alsger et al., 2018). Such strong evidences have shown that GTFS plays a critical role in supplementing other data sources. Different from AVL, APC and AFC data, GTFS data complies with industry standards regarding its data formats and allows for sharing as open-source data source. However, the complex and large-amount GTFS raw data cannot be used directly, and an effective and efficient data acquisition and processing solution is needed.

Predictive models: The ARIMA model, as one of the most popular statistics-based time series models, can find the best fit of the model to the past observations by using the Box–Jenkins approach (Box et al., 2015). For many years, the ARIMA model has been combined with artificial neural network (ANN) as hybrid solutions, which improve forecasting accuracy in linear and nonlinear modeling (Khashei and Bijari, 2011; Zhang, 2003). More recently, deep neural networks models such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Graph Neural Network (GNN) achieved great success for time series forecasting. Among them, RNN was designed to learn sequence data and has been widely used in different research areas, such as natural language processing (Mikolov et al., 2011), computer vision (Zheng et al., 2015), and game artificial intelligence games (Ha and Schmidhuber, 2018). It directly captures temporal correlations without learning trends and seasonality. A variant of RNN and Long Short-Term Memory (LSTM) was a commonly used and powerful tool for sequence prediction tasks (Hochreiter and Schmidhuber, 1997). A GTFS-based dataset provides useful information to study various types of train delays. A multi-scenario real-time train delay prediction model was proposed to evaluate the performance of the state-of-the-art machine learning methods using the GTFS-based dataset (Wu et al., 2021). Since this paper focuses on the development of the GTFS DAP framework, a variant of LSTM is employed as a benchmark delay prediction model to test the performance of the proposed framework.

3. Methodology

3.1. GTFS static and real-time data

GTFS data is entirely open and free, which consists of static and real-time formats (Google). GTFS static data defines a common format for public transport schedules with associated geographic information; while GTFS real-time data provides real-time information of public transport services, such as vehicle location and road congestion level. On the one hand, further investigation of GTFS static and real-time data creates opportunities to explore valuable information from daily operation data; on the other hand, such work serves as a foundation to develop and validate various models and algorithms to support planning and operation of public transport services. Fig. 1 depicts an example of a GTFS static about the relationship between files and entities. First, the GTFS static has two types of files: required files and optional files. The required files contain six txt files (in the red rectangle in Fig. 1), namely 'agency.txt', 'routes.txt', 'trips.txt', 'stop_times.txt', 'stops.txt', and 'calendar.txt'. All the other files are optional, such as 'shapes.txt' and 'calendar_dates.txt'.

GTFS real-time contains vehicle positions, service alerts, and trip updates. This study employed the trip updates feed to obtain delay information. A trip update for trip_id '600D.1384.124.128.T.8.57243995' that presents a train arriving at a station (stop_id: '2205114') 183 seconds late and leaving 194 seconds late, would look as follow:

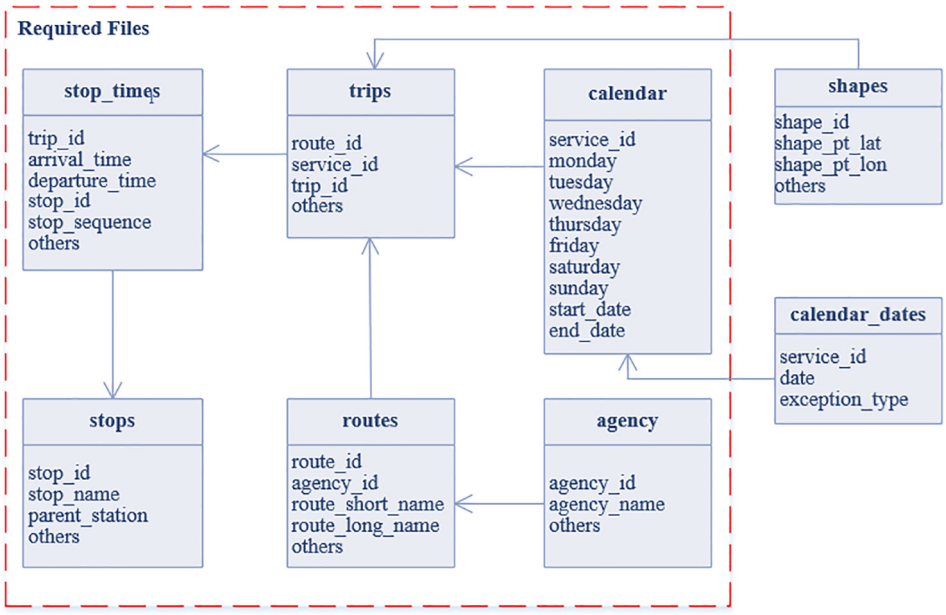

Fig. 1. GTFS file structure and entity relationship.

```
entity {
  id:" 600D.1384.124.128.T.8.57243995″
  trip_update{
    trip{
      trip_id: "600D.1384.124.128.T.8.57243995″
      schedule_relationship: SCHEDULED
      route_id: ""
    }
    stop_time_update{
      arrival{
        delay: 183
        }
      departure{
        delay: 194
      }
      stop_id: "2205114"
      schedule_relationship: SCHEDULED
    }
  }
}
```

3.2. GTFS static and real-time data merging and fusion

As a starting point, 'calendar_dates' text file is used as an example to create a usable GTFS timetable in comma-separated values (CSV) format. However, it is conditionally required that most GTFS data does not provide date information when a service exception does not occur, like all GTFS bundles from the Transport for NSW. In the 'calendar.txt' file, '1' denotes an available service, and '0′ implies that a service is not available for Monday to Sunday in the date range. We fill in missing information through using 'service_id' to create new data with regard to the date when the regular running of service occurs, namely 'scheduled_date'.

As shown in Table 1, we consider a relational schema R with attributes $atr(R)$, a functional dependency (FD) is defined as $->A$, wherein $X \subseteq atr(R)$ and $A \subseteq atr(R)$. We have X the left-hand side (LHS) and A the right-hand side (RHS). As the RHS of

Table 1
GTFS static data merging with real-time data.

Overlapped			Static			Real-time	
trip_id	stop_id	scheduled_date	arrival_time	departure_time	stop_name	arrival_delay	departure_delay
600D	202,291	2019/4/15	8:25:01	8:29:01	Bondi Junction Station	115	46
600D	202,761	2019/4/15	8:32:00	8:32:30	Edgecliff Station	71	77
600D	201,171	2019/4/15	8:34:30	8:35:00	Kings Cross Station	70	107
600D	2,000,361	2019/4/15	8:37:00	8:37:30	Martin Place Station	92	117
600D	2,000,394	2019/4/15	8:39:00	8:40:00	Town Hall Station	106	97
600D	2,000,345	2019/4/15	8:42:00	8:43:00	Central Station	75	67
600D	2,015,142	2019/4/15	8:44:36	8:45:06	Redfern Station	60	81
600D	2,205,114	2019/4/15	8:53:30	8:54:00	Wolli Creek Station	84	87
600D	2,220,364	2019/4/15	9:02:00	9:06:00	Hurstville Station	20	0

each FD is a subset of its LHS, the FD is trivial. Thus, agency_id$->$route_id and route_id, service_id$->$trip_id, stop_id are valid concerning a GTFS dataset instance.

A Trip Update API is called once every 10–30 s for extracting GTFS real-time data. All collected data are stored in CSV format. Firstly, the dataset contains a large number of repetitions, which are cleaned and filtered in the deduplication step. As the collected data is incremented based on the time stamp, 'trip_id' and 'stop_id' are used to keep the last one and remove the others for identifying and dropping duplicates, as shown in Table 1. Also, the GTFS real-time data online collection is demonstrated in Algorithm 1.

Algorithm 1
Online collection of GTFS real-time data

Input: entities E // collecting JSON objects with the API
Output: S // a set of entries is returned
Repeat
  For E **in** Feed_Entity
    If $E.hasField(\text{"}TU\text{"})$ // TU: trip update; VP: vehicle positions; SA: service alerts
      For *count* in range (length ($E.stop_1, \cdots E.stop_n$))
        If *E.stop. trip_id* and *E.stop. stop_id* exist
          $S.add(E.stop_{current})$
          $S.remove\left(E.stop_{previous}\right)$
        Else
          $S.add(E.stop_{current})$
      End for
    time.sleep(10)
    End for

Secondly, GTFS static (DB-S) and GTFS real-time (DB-R) can be fused as a unified dataset by mapping 'trip_id', 'stop_id', 'date' at the schema-based data fusion step. Hence, the schema matches should be DB-S.trip_id = DB-R.trip_id, DB-S.stop_id = DB-R.stop_id, and DB-S.scheduled_date = DB-R.Scheduled_Date. Thirdly, an outlier detection model expects the input data to be in the standard format, such as time and date, that need to be converted into the same unit (Ilyas and Chu, 2019). Herein, data transformation is applied to standardizing data format. An example of GTFS static and real-time fusion dataset containing main attributes is shown in Table 2.

Table 2
GTFS Static and Real-Time Data Fusion.

Trip_Number	Scheduled_Date	Arrival_Delay	Week_Index	Departure_Delay	Dwell_Time	Stop_Sequence	Running_Time
600D	2019/4/15	115	1	46	171	1	0
600D	2019/4/15	71	1	77	36	2	204
600D	2019/4/15	70	1	107	67	3	113
600D	2019/4/15	92	1	117	55	4	105
600D	2019/4/15	106	1	97	51	5	79
600D	2019/4/15	75	1	67	52	6	98
600D	2019/4/15	60	1	81	51	7	89
600D	2019/4/15	84	1	87	33	8	507
600D	2019/4/15	20	1	0	220	9	413

3.3. A GTFS data acquisition and preparation (DAP) framework

The proposed GTFS DAP framework can provide an interface to connect with a multimodal transport system and cloud services, as shown in Fig. 2. In a multimodal transport management system such as MaaS, an integrated data platform is used by data service providers, which is an intermediate layer between users and transport operators. It offers a data service bringing together all the modes of existing transportation. The provision of real-time information is shared by cloud computing services.

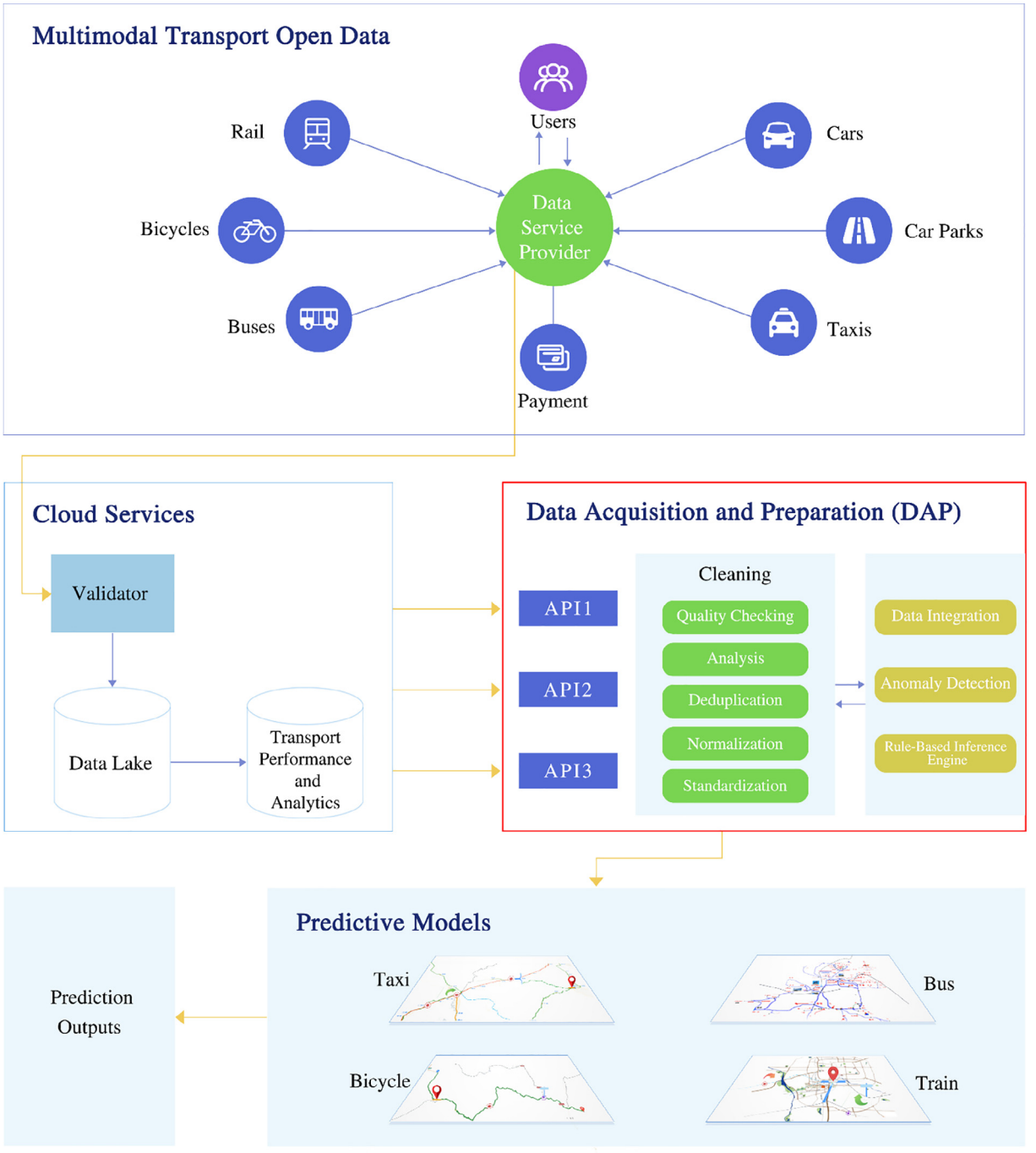

Fig. 2. A GTFS data acquisition and processing framework.

A validator validates data to ensure that there are no errors or issues before the data is loaded in the data lake of the cloud service. Moreover, data are uploaded or transferred from the validator into the data lake before any data is transferred to the Transport Performance and Analytics (TPA) database. The TPA converts credible transport data to the JSON format, which is easily shared and reanalyzed. In this study, the DAP was developed to collect data by using standard application programming interfaces (APIs).

Following data validation, data deduplication is implemented to remove incorrect or undesirable observation data. Then, the preprocessed GTFS real-time dataset is combined with GTFS static data to generate an integrated dataset, and the whole process can be regarded as data fusion. Furthermore, an anomaly detection method obtains predicted values, which are used to calculate the Z-scores for standardization to find abnormal data. Finally, the application of a rule-based inference engine can fill in missing values and provide additional useful information. Normalization is an essential data preprocessing step to scale features before training a prediction model. This process ensures that the scaling variables are within the same range of values to provide appropriate inputs to the predictive models. As a result, the proposed GTFS DAP framework can obtain multivariate time series datasets from the multimodal transport system platform.

As shown in Fig. 3, the data cleaning and aggregation tool is described as a workflow to depict the components of offline GTFS static, online GTFS real-time, data deduplication, schema-based data fusion, data transformation, outlier detection, and rule-based inference engine.

3.4. Anomalous timesteps detection

Delay prediction is generally divided into two main categories: short-term (real-time) and medium-/long-term. In terms of prediction tasks, the two modeling categories have different requirements for data imputation. For short-term or real-time prediction, we apply rules to identifying anomalies in the arrival or departure time in real-time. For the medium-term or long-term, we use predictive algorithms to find abnormal delay records from historical observation of a long time series.

Moreover, delay prediction can be regarded as the problem of time series analysis, which involves univariate and multivariate variants. As the multivariate method can automatically contain the univariate approach, traffic information prediction studies are conventionally referred as multivariate time series analysis. For time series forecasting, it is essential to filter out outliers to ensure that the observations are accurate and useful for modeling the subsequent prediction problem. Anomaly detection methods detect outliers, which are outside the scope of defining normal data. In this study, we apply a statistics-based outlier detection technique, the BSTS model (Brodersen et al., 2015). It is a state-space model (SSM) for time series data. The model can utilize posterior predictive samples to compute the posterior distribution of cumulative impact. Expected data points appear in high probability regions of the model, whereas outliers occur in the low probability regions of the model. Standardized residuals can be employed to detect outliers by comparing them to a Z-score. The formulation of the problem as a pair of equations is given by:

$$y_t = Z_t^T a_t + \varepsilon_t, \varepsilon_t \; N(0, \sigma^2) \tag{1}$$

$$a_{t+1} = T_t a_t + R_t \eta_t, \eta_t \; N(0, Q_t) \tag{2}$$

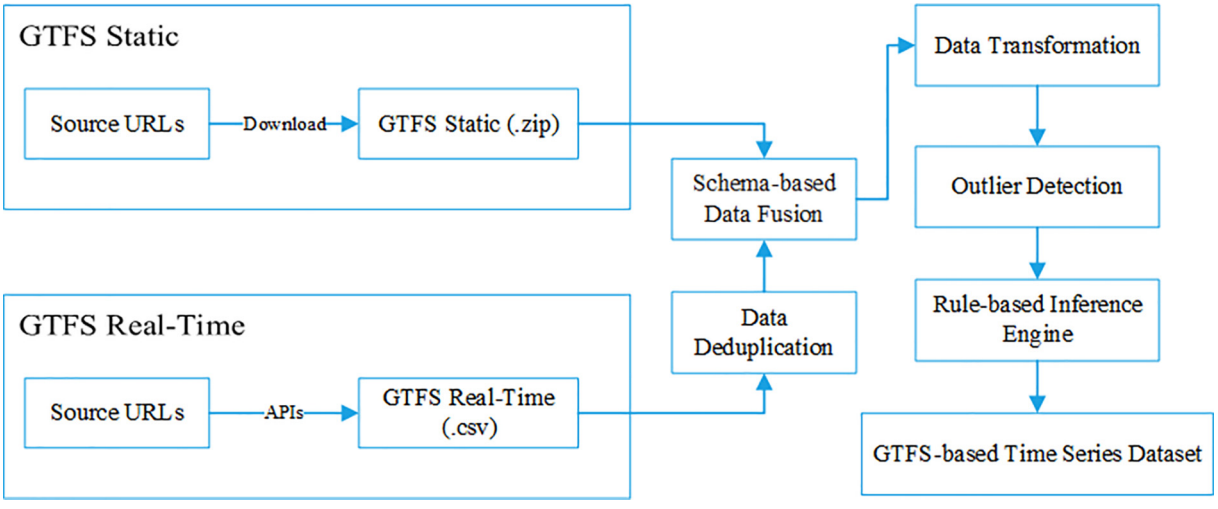

Fig. 3. Data cleaning and aggregation workflow.

wherein Eq. (1) is the observation model and Eq. (2) is the transition model (state equation). The transition model and observation model are both linear Gaussians. A linear Gaussian model is a Bayesian network, which defines multivariate Gaussian distributions.$\varepsilon\ N(0,\sigma^2)$ is an observation noise term. $\eta_t\ N(0,Q_t)$ is a transition noise term. y_t is observed data of the model at time t.a_t is a state vector. The transition model describes the evolution of the state vector from time step t to timestep $t+1$. Z_t^T is the observation matrix.T_t is a transition matrix.R_t is a control matrix. σ^2 and Q_t are covariance matrices.

Structural time series (STS) models contain three types of critical components: seasonality, regression, and local linear trends (Brodersen et al., 2015). For seasonality, we utilize date-time to generate a seasonal feature, which captures seasonal effects. The regression component is the essential state component of the model. For the local linear trend, an autoregressive (AR) model is applied to balance short-term information with previous steps, in which the value of each step is a noisy linear combination of the previous steps. The model can use the additive or the multiplicative form (Harvey, 1990). An STS model represents an observed time series as the additive form:

$$f(t)=\sum_{i=1}^{n} f_i(t)+\varepsilon_t \tag{3}$$

Thus, the observed time series can be decomposed into multiple components such as trend, seasonal patterns, and residual effects. n is the number of components. Eq. (3) can be expressed as the sum of components in the following form:

$$f(t)=f_1(t)+f_2(t)+..+f_n(t)+\varepsilon_t \tag{4}$$

For a given STS, each component is treated as a Bayesian model, which contains the set of model parameters θ and the observation vector $y=\{y_1,\cdots,y_t\}$. The posterior density $P(\theta|y)$ is computed by the Bayes' theorem:

$$P(\theta|y)=\frac{P(y|\theta)P(\theta)}{P(y)} \tag{5}$$

To learn the set of model parameters θ, the Kalman filter algorithm is utilized to derive the likelihood $P(y|\theta)$. Applying the Bayes' theorem to calculate the $P(\theta|y)$ is referred as an inference problem. Moreover, we can turn it into an optimization problem by minimizing the Kullback-Leibler (KL) divergence from $Q(\theta;\lambda)$ to $P(y|\theta)$.

$$Q(\theta;\lambda)^*=\arg\min_{\lambda}\mathrm{KL}(Q(\theta;\lambda)||P(\theta|y)) \tag{6}$$

where variational inference is applied to approximating the posterior distribution of the model parameters $P(\theta|y)$, by employing variational distribution $Q(\theta)$ with variational parameters λ from the given observation y. After the data y is observed, the Bayesian model has the marginal likelihood (model evidence):

$$P(y)=\int P(y,\theta)d\theta \tag{7}$$

As the marginal likelihood is a constant to λ, we can maximize the Evidence Lower Bound (ELBO) instead of minimizing the KL divergence:

$$Q(\theta;\lambda)^*=\arg\max_{\lambda} ELBO(Q(\theta)) \tag{8}$$

Furthermore, the gradient descent, as an optimization algorithm, is used to minimize $-ELBO(Q(\theta))$ regarding the variational parameters:

$$\nabla_\lambda ELBO(Q(\theta))=\nabla_\lambda\int Q(\theta)\log\frac{P(y,\theta)}{Q(\theta)}d\theta \tag{9}$$

To compute integration, sampling is often exploited in the process. Hence, we incorporate the observations directly into the model by applying a reparameterization gradient (Kingma and Welling, 2014; Rezende et al., 2014). Moreover, we firstly use multiple attributes as inputs. Then, the model is trained to maximize $ELBO(Q(\theta))$. Training a BSTS with rule-based inference engine for data preprocessing is indicated in Algorithm 2. Although advanced deep learning models can derive approximate values very well, they fail to preprocess GTFS data. GTFS data does not directly records the actual information of running trains, such as actual arrival time, actual departure time, actual running time, and actual dwell time, which are essential for real-time prediction and long-term prediction. Considering the causal relationships among various GTFS variables, we propose a rule-based inference engine to derive more useful variables accurately from existing variables by using the scheduled arrival time A_{ts}, scheduled departure time D_{ts}, actual arrival delay AAD_{ta}, and actual departure delay ADD_{ta} at time step t.

Algorithm 2
Training a BSTS with rule-based inference engine

Require: type of tasks *TS*, spatiotemporal attributes y_t^N, the number of attributes *N*, random variables *x*, variational distribution $Q(\theta)$.
Real-time input data: actual arrival time $A_{\boldsymbol{t}a}$, actual departure time $D_{\boldsymbol{t}a}$, actual running time $R_{\boldsymbol{t}a}$, and actual dwell time $W_{\boldsymbol{t}a}$.
If *TS* = Preprocessing a data set is used for a real-time forecast:
  For $\boldsymbol{n} = 1 \rightarrow \boldsymbol{i}$ do
    $A_{\boldsymbol{t}a} = A_{\boldsymbol{t}s} + AAD_{\boldsymbol{t}a}$
    $D_{\boldsymbol{t}a} = D_{\boldsymbol{t}s} + ADD_{\boldsymbol{t}a}$
    If $\left(A_{\boldsymbol{t}a} - A_{(\boldsymbol{t}-1)_a}\right) \leq 0$ or $\left(D_{\boldsymbol{t}a} - D_{(\boldsymbol{t}-1)_a}\right) \leq 0$:
      Set anomalies to *NaN*
      Applying a real-time delay prediction model
Else if *TS* = Preprocessing a data set is used for a long-term forecast:
  Initialize λ, *N*, $t = 1$.
  Output: λ
    Repeat
      Draw *S* samples from the variational approximation
      For s = 1 **to** *S* **do**
        x_s $Q(\theta)$
      End for
      Estimate the gradient $\nabla_\lambda ELBO(Q(\theta)) = \nabla_\lambda \int Q(\theta) \log \frac{P(y_t^N, \theta)}{Q(\theta)} d\theta$ as in equation (9)
      Set the learning rate
      Update the parameter λ
      $t = t + 1$
    Until the convergence conditions are satisfied
    If z-score > threshold value
      Set anomalies to *NaN*
      Applying a data imputation algorithm
Applying a rule-based inference engine to search more useful variables, including $A_{\boldsymbol{t}a}$, $D_{\boldsymbol{t}a}$, $R_{\boldsymbol{t}a}$, and $W_{\boldsymbol{t}a}$
For each stop $es = (A_{\boldsymbol{t}a}, D_{\boldsymbol{t}a}, R_{\boldsymbol{t}a}, W_{\boldsymbol{t}a})$ **do**
    $A_{\boldsymbol{t}a} = A_{\boldsymbol{t}s} + AAD_{\boldsymbol{t}a}$
    $D_{\boldsymbol{t}a} = D_{\boldsymbol{t}s} + ADD_{\boldsymbol{t}a}$
    $R_{\boldsymbol{t}a} = A_{\boldsymbol{t}a} - D_{\boldsymbol{t}-1a}$
    $W_{\boldsymbol{t}a} = D_{\boldsymbol{t}a} - A_{\boldsymbol{t}a}$
End for

3.5. Multivariate multistep delay prediction

Delay prediction can be expressed as a task extract information from historical data to accurately estimate future delay times for a corresponding mode of transport. In this study, a multivariate multistep LSTM (MM-LSTM) model is trained to verify the effectiveness of the proposed GTFS DAP framework. As indicated in Fig. 4, an LSTM has the short-term state h_{t-1} and the long-term state from c_{t-1} to c_t. Additionally, the input X^t is fed to sigmoid functions σ and an activation function *tanh*.

Firstly, the LSTM drops some memories in the forget gate and adds some new memories to the input gate by applying the addition operation. Secondly, the long-term state is passed through the activation function. Lastly, the output gate filters the results to generate h_t. W_{xi}, W_{xf}, W_{xc}, W_{xo}, W_{hi}, W_{hf},W_{hc}, and W_{ho} denote the weight matrices that connect the input and the hidden vectors to the corresponding gates, respectively.b_i,b_f, b_c, and b_o express bias vectors. The equations of LSTM are demonstrated in Eq. (10), wherein the operator '$\circ$' is the Hadamard product.

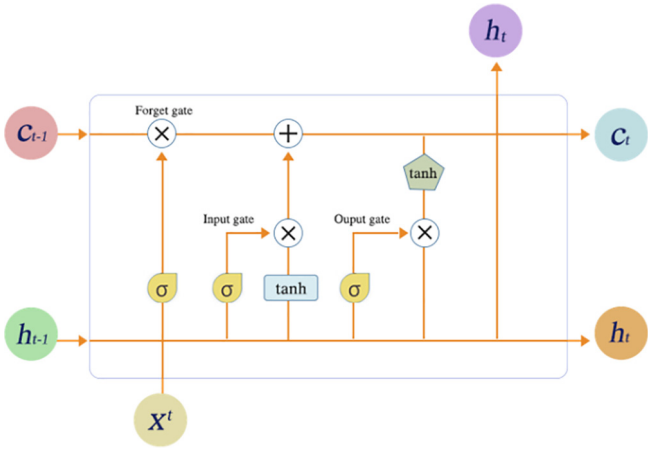

Fig. 4. Illustration of an LSTM network.

$$
\begin{aligned}
i_t &= \sigma\left(W_{xi}\, X^t + \; W_{hi} h_{t-1} + W_{ci}^{\circ} c_{t-1} + b_i\right) \\
f_t &= \sigma\left(W_{xf}\, X^t + \; W_{hf} h_{t-1} + W_{cf}^{\circ} c_{t-1} + b_f\right) \\
c_t &= f_t^{\circ} c_{t-1} + i_t^{\circ} tanh\left(W_{xc} X^t + \; W_{hc} h_{t-1} + b_c\right) \\
o_t &= \sigma\left(W_{xo}\, X^t + \; W_{ho} h_{t-1} + W_{co}^{\circ} c_t + b_o\right) \\
h_t &= o_t^{\circ} tanh(c_t)
\end{aligned}
\tag{10}
$$

4. Numerical experiments

4.1. Data description

To validate the proposed DAP framework, we choose train services in Sydney as a case study. For operational data, we utilize Sydney Trains' GTFS static and real-time data obtained from the Transport for NSW's open data portal, which publishes its data regularly (Transport for NSW). Spatial-temporal feature analysis is performed on a fused dataset of the GTFS static and real-time data, which consists of 150-day data observations of T4 Eastern Suburbs and Illawarra (ESI) train line from April 15 to November 8 in 2019.

In spatial dimension, the T4 line has four routes, including ESI_1a (Bondi Junction to Waterfall), ESI_1d (Bondi Junction to Cronulla), ESI_2a (Waterfall to Bondi Junction), and ESI_2d (Cronulla to Bondi Junction). Fig. 5 shows cumulative sums of

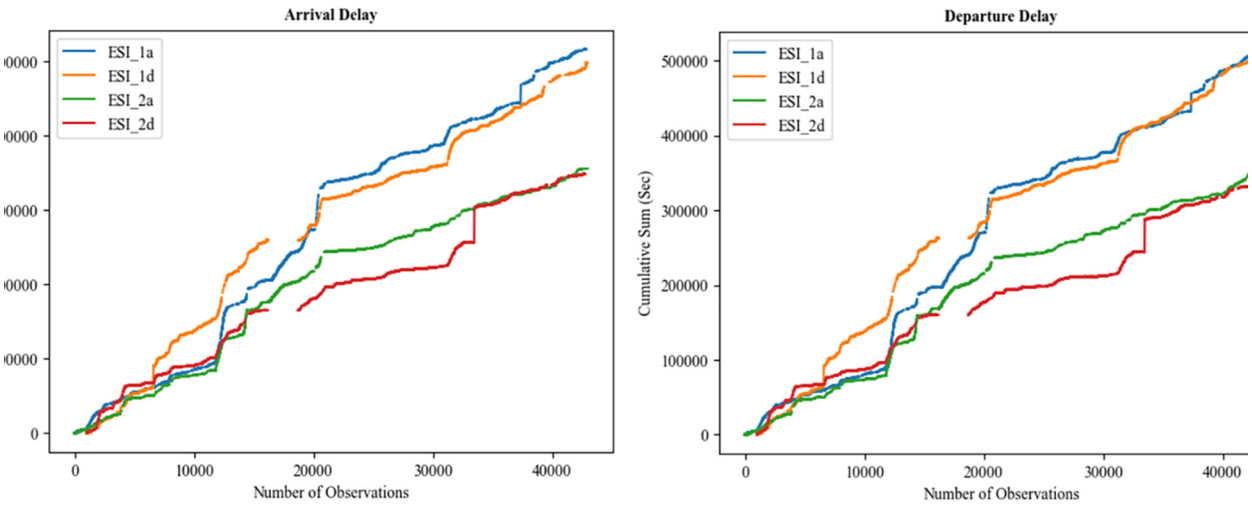

Fig. 5. Cumulative train delay on T4 ESI train line.

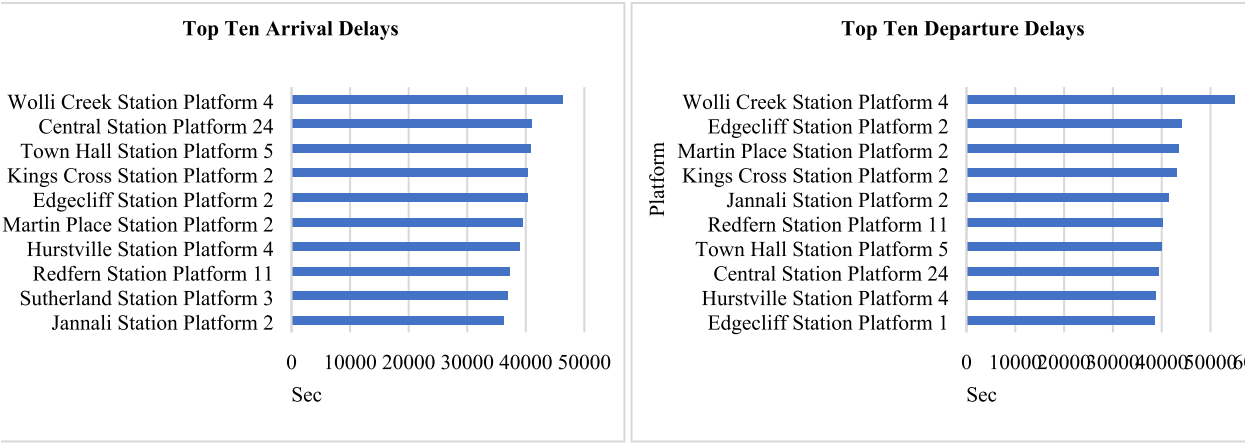

Fig. 6. Top ten arrival and departure delays.

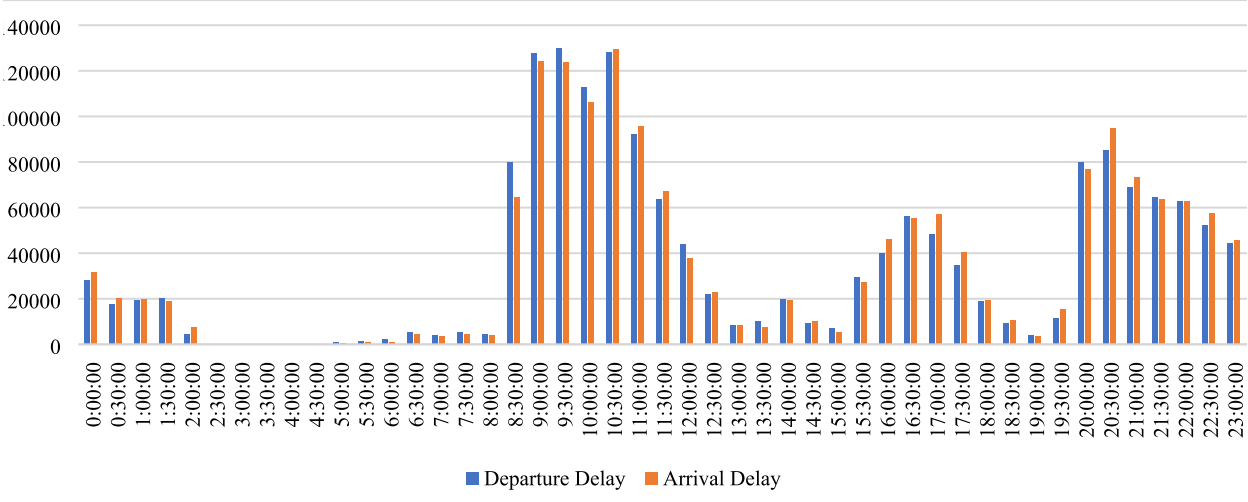

Fig. 7. Distribution of train delay at different time segments.

train delay of the four routes on the T4 line based on 42,946 observations at discrete time series points. For departure delays, ESI_1a and ESI_1d have no noticeable difference. However, ESI_1a has the most significant arrival delays. In addition, the top ten arrival delay platforms and top ten departure delay platforms are listed as shown in Fig. 6. It indicates that Wolli Creek station platform 4 has the most significant delays on arrival and departure trains.

As observed from Fig. 7, for time-interval data in the temporal dimension, it is clear that delay peaks occur at three different periods: A.M. peak from 8:30 am to 11:30 am, P.M. peak from 3:30 pm to 5:30 pm, and Night Inter-peak from 8:00 pm to 11:00 pm. In addition, the A.M. peak period has more than double the delay time of the P.M. peak period. In this study, we aim to investigate the highest delay trips from T4. Thus, we choose trips after 8:30 am and passing through the Wolli Creek Station Platform 4. Furthermore, it can be found that a trip number '600D' of the ESI_1a meets the corresponding conditions as a qualified sample dataset. The selected experimental dataset consists of 150-day data observations at seven stations. Its daily scheduled arrival time is 8:34:30 am, and the expected departure time is 8:35:00 am at Bondi Junction Station from Monday to Friday morning.

For details of trip '600D', a specific train line connecting Bondi Junction Station and Hurstville Station, namely BJS-HS, is selected for numerical experiments. The BJS-HS line is one of the busiest urban rail lines in Sydney, linking CBD and some of the most populous suburbs and also localities around the airport. Among them, Hurstville station has the highest number of daily transit patrons outside Sydney's CBD, while Bondi Junction connects to the most popular beach in the southern hemisphere. Trains operating on this line pass through seven major stations, including Edgecliff Station (ES), Kings Cross Station (KCS), Martin Place Station (MPS), Town Hall Station (THS), Central Station (CS), Redfern Station (RS), and Wolli Creek Station (WCS), as shown in Fig. 8.

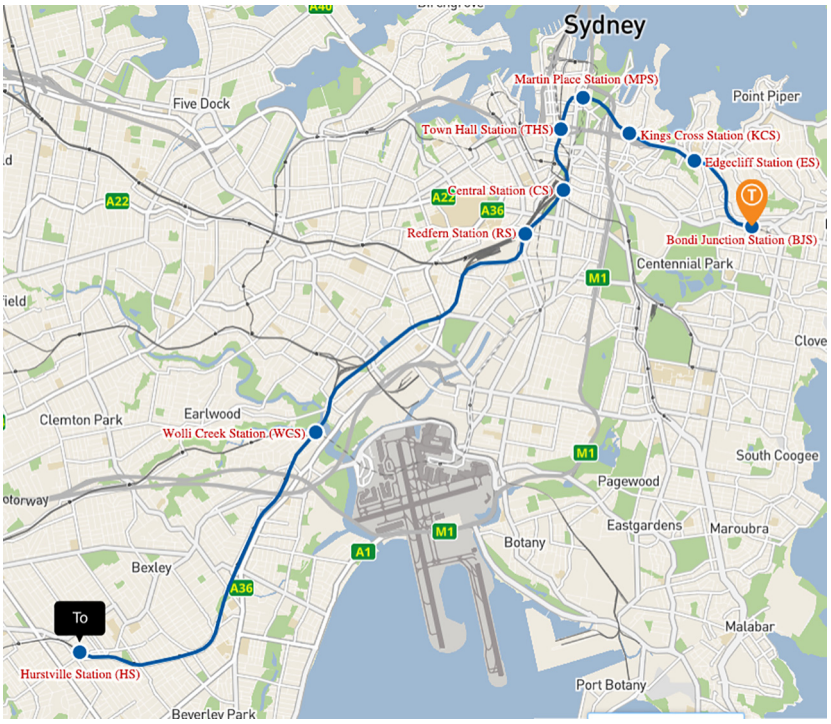

Fig. 8. Train line BJS-HS in Sydney.

4.2. Experimental setup

According to Algorithm 2, we have two ways to deal with anomalies: One is to set anomalies to *NaN* (non-imputed), and the other is to use a data imputation algorithm (imputed). This study conducts a comparison between a non-imputed strategy and an imputed strategy for the long-term forecasting of train delays. The imputed strategy is to apply the mean of observed data for replacing missing values for arrival delays and departure delays in the GTFS real-time data before they are used for modeling traffic information prediction tasks. Additionally, the rule-based inference engine creates more useful variables from existing variables. Moreover, the BSTS model constructs one-step-ahead predictive distributions for all time-steps using samples from the variational posterior. To detect anomalous time steps in time series data, we use Z-scores to classify the observed values into two subgroups to identify whether a value is within a defined predictive interval.

$$Z = \frac{|x - \bar{x}|}{s} \tag{11}$$

wherein Z represents a standard score, x is the observed value, $\bar{x}$ denotes the mean of the predictive distributions, and s is the standard deviation of the predictive distributions. A time-series data point would be considered abnormal if its Z-score is greater than a threshold value. As a result, according to the Z-score (the standard normal) table, we identify the corresponding data points, which contain regular delay patterns.

The LSTM uses the first 105 days of historical records as the training set, the next 30 days as the validation set, and the last 15 days as the testing set, respectively.

4.3. Evaluation metric

In the experiments, four metrics are applied to evaluating the prediction performance of the different algorithms combined with the proposed method, namely, Root Mean Square Error (RMSE), Mean Absolute Error (MAE), Median Absolute Error (MedAE), and Symmetric Mean Absolute Percentage Error (SMAPE). SMAPE provides a result between 0% and 200%. $\hat{y}_i$ and y_i denote the predicted and actual values, respectively. The definitions of the four metrics are written as follows.

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}|\hat{y}_i - y_i| \tag{12}$$

Table 3
Arrival delay and departure delay prediction performances.

Model	Station	RMSE (sec)		MAE (sec)		MedAE(sec)		SMAPE (%)	
		Arrival	Departure	Arrival	Departure	Arrival	Departure	Arrival	Departure
Non-Imputed	ES	48.4	53	42	45	88.4	46.1	169.2	131.2
	KCS	54.5	54.9	47.3	47.2	45.2	48.3	168.9	114.6
	MPS	63.8	77.7	53.9	63.6	47.2	65	167	121.3
	THS	65.7	76.6	55.5	56.9	147.6	43.5	137.8	102.6
	CS	65.6	61.8	51.7	41	46.9	32.4	153.2	151.7
	RS	63.5	77.3	46.5	55.7	28.7	45.7	167.3	156.5
	WCS	51.9	55.5	45.8	48.8	47.6	57.5	128	118.5
Imputed	ES	23.6	48	13.4	38.8	1.8	25.4	24.5	57.3
	KCS	21.7	39.5	15.6	27.9	7.9	18.4	27.9	40.9
	MPS	35.8	51.4	27.3	40.6	22.4	39.2	35.3	47.4
	THS	47.1	53.3	34.3	37.6	24.5	22.8	36.1	55.6
	CS	38.1	50.1	24.6	33.7	13.2	16.3	33.6	39
	RS	40.9	39.8	25.8	23.9	11.4	12.1	35	23.8
	WCS	49.1	55.9	32.1	36.9	12.1	12.3	51.4	49

$$\text{RMSE} = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2} \tag{13}$$

$$\text{MedAE} = median(|\hat{y}_1 - y_1|, \cdots, |\hat{y}_i - y_i|) \tag{14}$$

$$\text{SMAPE} = \frac{200\%}{m}\sum_{i=1}^{m}\frac{|\hat{y}_i - y_i|}{|\hat{y}_i| + |y_i|} \tag{15}$$

4.4. Model comparison

With detailed numerical results, Table 3 shows the performance of our proposed framework in predicting traffic information over the entire data set with the chosen testing LSTM. An Imputed dataset can achieve the best performance in RMSE, MAE, MedAE, and SMAPE across nearly all stations. When the Imputed dataset is used for prediction, the performance is better than the Non-Imputed dataset. According to the prediction results, the rule-based inference engine is an effective strategy for pre-processing the entire data set.

Fig. 9 presents a comparison of RMSE, MAE, and MedAE errors produced by Non-Imputed and Imputed datasets for arrival and departure delay predictions. Due to the complexity and noise in the GTFS real-time data, the data pre-processing can identify and remove most of the impact of outliers from the raw data. Additionally, the data pre-processing can also reduce the impact of missing values more effectively. It can be seen from the Fig. 9 that the Non-Imputed has two prediction anomalies (blue points) at MedAE. As a result, the Non-Imputed indicates large prediction errors among the three error metrics.

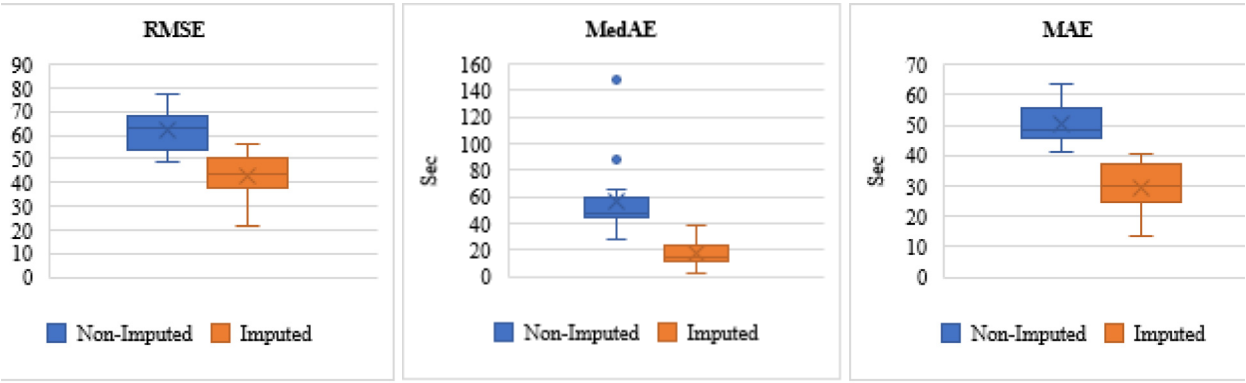

Fig. 9. RMSE, MAE, and MedAE for arrival and departure delays.

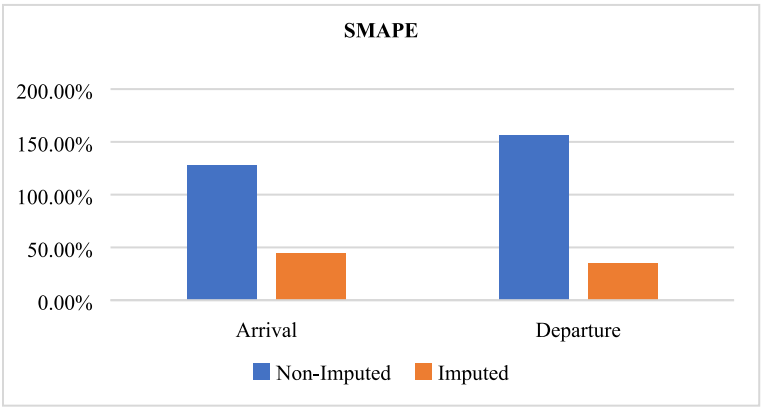

Fig.10. SMAPE of the predictions.

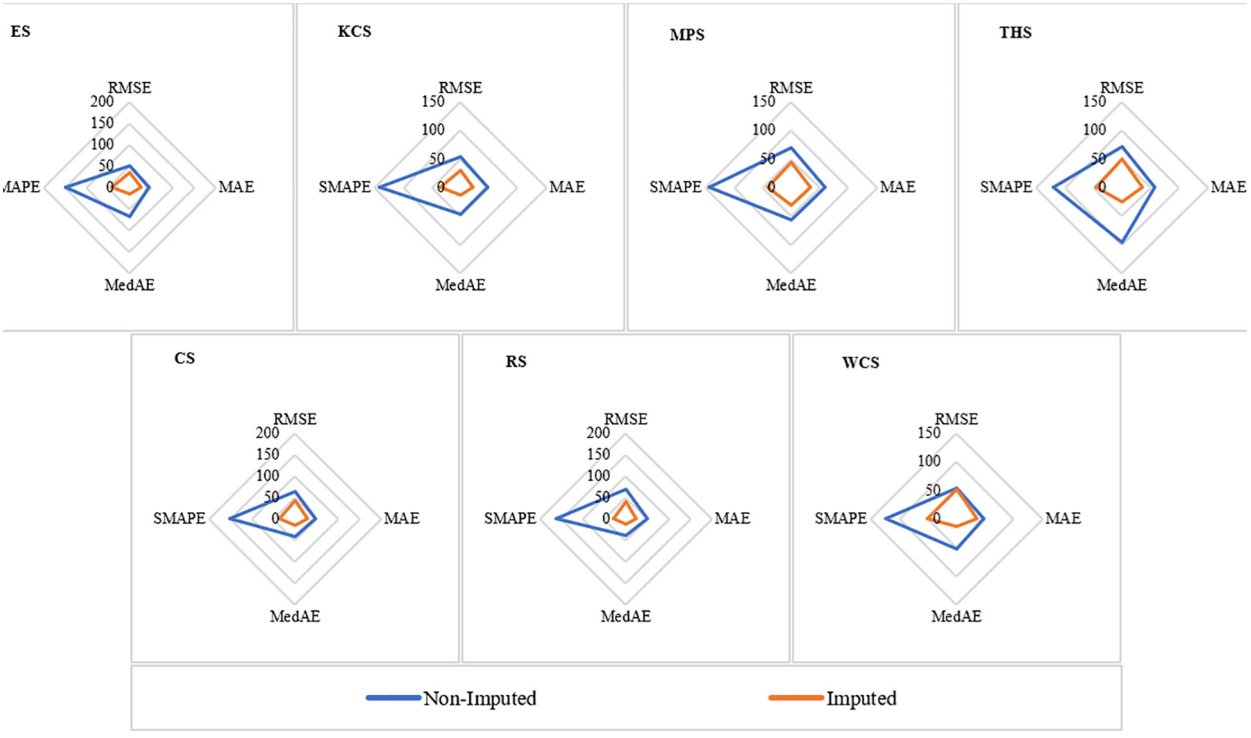

Fig. 11. Performance evaluation at selected train stations.

Furthermore, the imputed strategy indicates better performance than other models in terms of the maximum, minimum, and median errors. The error distribution of the Imputed is more concentrated in the distribution of errors. In contrast, the Non-Imputed has relatively larger errors.

Fig. 10 shows that the Imputed has the lowest percentage of errors for the arrival delay and departure delay at SMAPE. Additionally, Non-Imputed has the lowest accuracy in terms of SMAPE. In summary, it can be seen that the Imputed dataset provides a sufficiently good prediction performance in terms of validation metrics.

To validate the robustness of the proposed framework, we further compare the performance of the proposed model on Non-Imputed datasets and Imputed datasets for selected train stations, as shown in Fig. 11. In general, the proposed method produces less errors and shows a more stable prediction ability among four error indexes. According to the results, using the imputed strategy can effectively develop a DAP framework to produce a GTFS integrated dataset. After identifying the best combination, all missing values can be replaced by the imputed strategy for the arrival delay and departure delay. With minimized anomaly, these datasets can be used to feed a diversity of deep learning models.

5. Conclusion

This paper proposed and implemented a DAP framework to process the GTFS data in both static and real-time formats. Moreover, a comprehensive data cleaning workflow was proposed. An MM-LSTM model was applied to exploring the Non-imputed and Imputed datasets for predicting delay information in the numerical experiments.

Overall, even though the GTFS data would be complex and noise-intensive, our proposed method could create a high-quality time series dataset for traffic information forecasting with deep learning algorithms. The new method and tool can also be used to solve the real-time delay prediction tasks, including regular and irregular delays. It is challenging to obtain delay records caused by abnormal events. GTFS data provides a unique opportunity to obtain such valuable information. Data quality is the basis of data-driven modeling. However, such a work was overlooked in literature and the proposed GTFS DAP framework in this paper aimed to fill the gap.

The fused GTFS data is useful for traffic information forecasting and generates great potential to complement other data sources, such as APC and AFC. Furthermore, GTFS real-time Vehicle Position and GTFS real-time Service Alerts can be fused by modifying our proposed DAP framework. Since the primary purpose of the paper is to design a practical and ready-to-use DAP framework, some other methods, such as Hamiltonian Monte Carlo for outlier detection and Generative Adversarial Network for data imputation, have not been involved. These models should be evaluated and compared in future studies.

Conflict of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Alsger, A., Assemi, B., Mesbah, M., Ferreira, L., 2016. Validating and improving public transport origin-destination estimation algorithm using smart card fare data. Transp. Res. Part C: Emerg. Technol. 68, 490–506.
Alsger, A., Tavassoli, A., Mesbah, M., Ferreira, L., Hickman, M., 2018. Public transport trip purpose inference using smart card fare data. Transp. Res. Part C: Emerg. Technol. 87, 123–137.
Barbeau, S.J., 2018. Quality control-lessons learned from the deployment and evaluation of GTFS-realtime feeds. 97th Annual Meeting of the Transportation Research Board, Washington, DC. Available at <https://trid.trb.org/view/1496848>.
Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M., 2015. Time Series Analysis: Forecasting and Control. John Wiley & Sons.
Brakewood, C., Macfarlane, G.S., Watkins, K., 2015. The impact of real-time information on bus ridership in New York City. Transp. Res. Part C: Emerg. Technol. 53, 59–75.
Brodersen, K.H., Gallusser, F., Koehler, J., Remy, N., Scott, S.L., 2015. Inferring causal impact using Bayesian structural time-series models. Ann. Appl. Stat. 9, 247–274.
Du, B., 2019. Estimating travellers' trip purposes using public transport data and land use information. The Tenth Triennial Symposium on Transportation Analysis (TRISTAN X).
Du, B., Dublanche, P.-A., 2018. Bus bunching identification using smart card data. In: 2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS). IEEE, pp. 1087–1092.
Gaudette, P., Chapleau, R., Spurr, T., 2016. Bus network microsimulation with general transit feed specification and tap-in-only smart card data. Transp. Res. Rec. 2544, 71–80.
Giraud, A., Trépanier, M., Morency, C., Légaré, F., 2016. Data fusion of APC, smart card and GTFS to visualize public transit use (No.CIRRELT-2016-54). CIRRELT, Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation. Available at: <https://www.cirrelt.ca/documentstravail/cirrelt-2016-54.pdf>,[Accessed 18 December 2021].
Gong, Z., Du, B., Liu, Z., Zeng, W., Perez, P., Wu, K., 2020. SD-Seq2seq: A deep learning model for bus bunching prediction based on smart card data. The 29th International Conference on Computer Communications and Networks (ICCCN), Honolulu, Hawaii, USA.
Google, Google Transit APIs. Google, Inc. Available at: <https://developers.google.com/transit>, [Accessed 18 December 2021].
Gupta, M., Gao, J., Aggarwal, C.C., Han, J., 2013. Outlier detection for temporal data: A survey. IEEE Trans. Knowl. Data Eng. 26 (9), 2250–2267.
Ha, D., Schmidhuber, J., 2018. Recurrent world models facilitate policy evolution. Advances in Neural Information Processing Systems, 2450–2462.
Harvey, A.C., 1990. Forecasting, Structural Time Series Models and the Kalman Filter. Cambridge University Press.
Hill, D.J., Minsker, B.S., 2010. Anomaly detection in streaming environmental sensor data: A data-driven modeling approach. Environ. Modell. Software 25 (9), 1014–1022.
Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.
Huang, Z., Ling, X., Wang, P.u., Zhang, F., Mao, Y., Lin, T., Wang, F.-Y., 2018. Modeling real-time human mobility based on mobile phone and transportation data fusion. Transp. Res. Part C: Emerg. Technol. 96, 251–269.
Ilyas, I.F., Chu, X., 2019. Data Cleaning. Association for Computing Machinery.
Jammalamadaka, S.R., Qiu, J., Ning, N., 2018. Multivariate Bayesian structural time series model. J. Mach. Learn. Res. 19, 2744–2776.
Johnson, M., Willsky, A., 2014. Stochastic variational inference for Bayesian time series models. Int. Conf. Mach. Learn., 1854–1862
Khashei, M., Bijari, M., 2011. A novel hybridization of artificial neural networks and ARIMA models for time series forecasting. Appl. Soft Comput. 11 (2), 2664–2675.
Kingma, D.P., Welling, M., 2014. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*.
Kumar, P., Khani, A., He, Q., 2018. A robust method for estimating transit passenger trajectories using automated data. Transp. Res. Part C: Emerg. Technol. 95, 731–747.
Kusakabe, T., Asakura, Y., 2014. Behavioural data mining of transit smart card data: a data fusion approach. Transp. Res. Part C: Emerg. Technol. 46, 179–191.
Le, N.D., Martin, R.D., Raftery, A.E., 1996. Modeling flat stretches, bursts outliers in time series using mixture transition distribution models. J. Am. Stat. Assoc. 91 (436), 1504–1515.
Mahrsi, M.K.E., Come, E., Oukhellou, L., Verleysen, M., 2017. Clustering smart card data for urban mobility analysis. IEEE Trans. Intell. Transp. Syst. 18 (3), 712–728.
McCallum, Q.E., 2012. Bad Data Handbook: Cleaning Up the Data So You Can Get Back to Work. O'Reilly Media, Inc.
Mikolov, T., Kombrink, S., Burget, L., Černocký, J., Khudanpur, S., 2011. Extensions of recurrent neural network language model. In: 2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). IEEE, pp. 5528–5531.

Nassir, N., Khani, A., Lee, S.G., Noh, H., Hickman, M., 2011. Transit stop-level origin-destination estimation through use of transit schedule and automated data collection system. Transp. Res. Rec. 2263, 140–150.
Pelletier, M.-P., Trépanier, M., Morency, C., 2011. Smart card data use in public transit: A literature review. Transp. Res. Part C: Emerg. Technol. 19, 557–568.
Peters, J., Janzing, D., Schölkopf, B., 2017. Elements of Causal Inference. The MIT Press.
Rezende, D.J., Mohamed, S., Wierstra, D., 2014. Stochastic backpropagation and approximate inference in deep generative models. In: Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32. JMLR.org, Beijing, China, pp. II–1278–II–1286.
Scott, S.L., Varian, H.R., 2014. Predicting the present with Bayesian structural time series. Int. J. Math. Modell. Numer. Optimis. 5, 4–23.
Scott, S.L., Varian, H.R., 2015. Bayesian Variable Selection for Nowcasting Economic Time Series. Economic Analysis of the Digital Economy, p. 119.
Tang, L., Thakuriah, P.(., 2012. Ridership effects of real-time bus information system: A case study in the City of Chicago. Transp. Res. Part C: Emerg. Technol. 22, 146–161.
Transport for NSW, General Transit Feed Specification (GTFS) and GTFS-Realtime (GTFS-R). Available at: <https://opendata.transport.nsw.gov.au/documentation>, [Accessed 18 December 2021].
Tsay, R.S., Pena, D., Pankratz, A.E., 2000. Outliers in multivariate time series. Biometrika 87, 789–804.
Wang, Y., Zhang, D., Liu, Y., Dai, B.o., Lee, L.H., 2019. Enhancing transportation systems via deep learning: a survey. Transp. Res. Part C: Emerg. Technol. 99, 144–163.
Wu, J., Wu, Q., Shen, J., Cai, C., 2020. Towards attention-based convolutional long short-term memory for travel time prediction of bus journeys. Sensors 20, 3354.
Wu, J., Zhou, L., Cai, C., Dong, F., Shen, J., Sun, G., 2019. Towards a general prediction system for the primary delay in urban railways. In: 2019 IEEE Intelligent Transportation Systems Conference (ITSC). IEEE, pp. 3482–3487.
Wu, J., Wang, Y., Du, B., Wu, Q., Zhai, Y., Shen, J., Zhou, L., Cai, C., Wei, W., Zhou, Q., 2021. The Bounds of Improvements Toward Real-Time Forecast of Multi-Scenario Train Delays. IEEE Transactions on Intelligent Transportation Systems 1–12. https://doi.org/10.1109/TITS.2021.3099031.
Zeng, W., Fu, C.-W., Arisona, S.M., Erath, A., Qu, H., 2014. Visualizing mobility of public transportation system. IEEE Trans. Visual Comput. Graphics 20, 1833–1842.
Zhang, G.P., 2003. Time series forecasting using a Hybrid ARIMA and neural network model. Neurocomputing 50, 159–175.
Zhao, Z., Koutsopoulos, H.N., Zhao, J., 2020. Discovering latent activity patterns from transit smart card data: a spatiotemporal topic model. Transp. Res. Part C: Emerg. Technol. 116, 102627.
Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., Torr, P.H., 2015. Conditional random fields as recurrent neural networks. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 1529–1537.
Zheng, Y.u., Capra, L., Wolfson, O., Yang, H., 2014. Urban computing: concepts, methodologies, and applications. ACM Trans. Intell. Syst. Technol. (TIST) 5 (3), 1–55.
Zhu, L., Yu, F.R., Wang, Y., Ning, B., Tang, T., 2019. Big data analytics in intelligent transportation systems: a survey. IEEE Trans. Intell. Transp. Syst. 20 (1), 383–398.