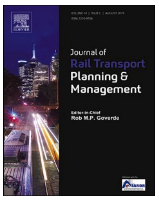
Journal of Rail Transport Planning & Management 34 (2025) 100518

Contents lists available at ScienceDirect

Journal of Rail Transport Planning & Management

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



Predicting primary delay of train services using graph-embedding based machine learning

Ruifan Tang [*], Ronghui Liu, Zhiyuan Lin

Institute for Transport Studies, University of Leeds, 34-40 University Rd, Leeds, LS2 9JT, United Kingdom

A R T I C L E I N F O

Keywords:
Machine learning
Structural deep network embedding
Train delay prediction
Graph embedding
Singular value decomposition
Principle component analysis

A B S T R A C T

Train delays can cause huge economic loss and passenger dissatisfaction. The Train Delay Prediction Problem has been investigated by a large number of studies. How to best represent certain features of a train is key to successful prediction. For instance, due to its complex topological nature, a train's route (i.e., origin, intermediate stations and destination) is one of the most difficult features to effectively represent. This study introduces graph embedding to understand and model the complex structure of a railway network which is able to capture a comprehensive collection of features including network topology, infrastructure and train profile. In particular, for the first time, we propose an approach to embed a train's route in a network topology perspective based on Structural Deep Network Embedding (SDNE) and Singular Value Decomposition (SVD). Compared to a conventional advanced method, Principle Component Analysis (PCA), our route embedding not only significantly reduces feature vector length and computational effort, but is also highly accurate and reliable in terms of capturing network topology as evidenced by K-means clustering. Computational experiments based on real-world cases from a UK train operator (TransPennine Express) show our graph-embedding based models are competitive in prediction accuracy and F1-score while are substantially computationally efficient compared to PCA.

1. Introduction

1.1. Background

Delays on rail networks can cause significant disruptions to the whole rail system, creating challenges not only for passengers, operators, and also the broader transportation infrastructure. Beyond passenger subsequent journeys will be missed, these delays lead to revenue losses for operators and increase operational costs due to unplanned adjustments in train scheduling and resource allocation. In the 2019/20 pre-pandemic period, there were 5.5 million delay compensation payments to passengers – that is for delays over 30 min including cancellations (ORR, 2020).[1] In dense urban networks, a single delay incident can quickly propagate through the neighboring tracks, lines and stations, to a wider network, further amplifying the problem (Ghaemi et al., 2017).

Many studies have been conducted on Train Delay Prediction (TDP) to improve the dependability and quality of railway passenger services and reduce financial losses for passengers and service providers. Delay prediction in the railway sector is a process of

* Corresponding author. University of Leeds, 34University Rd, Leeds, LS2 9JT, United Kingdom.
E-mail address: ml18r22t@leeds.ac.uk (R. Tang).

[1] https://dataportal.orr.gov.uk/media/1658/delay-compensation-claims-factsheet-2019-20-q2.pdf.

https://doi.org/10.1016/j.jrtpm.2025.100518
Received 5 May 2023; Received in revised form 27 February 2025; Accepted 28 February 2025
Available online 6 March 2025

estimating probabilities that a train fail to arrive at a subsequent check point (i.e., stations or junctions) on the time shown on a pre-defined timetable (Wen et al., 2020). A delay can be measured in terms of the differences between scheduled arrival (departure) time and actual arrival (departure) times, termed as 'arrival delays' ('departure delay'). Furthermore, train delays can be classified as primary delays and secondary delays. Primary delays are caused by issues that directly affect the infrastructure or operation of the railway system, and one essential feature of primary delays is that they are only relevant to the occurrence of unexpected events, regardless of the train's punctuality prior to the event (Rößler et al., 2021). Secondary delays are derived from the propagation of primary delays from the current location to the adjacent check point. In terms of quantity, primary delay is the most common type in a general railway system, and it has received significant attention in delay management from both train operators and passengers. In the context of this study, we limit our scope within primary delay prediction. Specifically, it concentrates on delay prediction for a given service on a given route. This prediction is based on historical timetable data, primary delays experienced on the same route, and railway network dependencies that services hold during their journeys. It does not involve real-time train delay control, rescheduling, or real-time delay calculation.

The counterpart of real-time primary delay observed at intermediate stations is the concept of 'generalized primary delay', which considers the entire running route and all stations on the route as a whole. Generalized primary delay is defined as the cumulative delay encompassing all primary delays occurring along the route. Within our context, generalized delay is formulated as a measure categorizing a given service into one of five delay levels: (0) for no delay, (1) mild delay, (2) moderate delay, (3) serious delay, and (4) severe delay. With this consideration, our goal is to identify an appropriate method to prognosticate the generalized primary delay for a specific service on a particular route by using historical primary delays encountered on that route, enables railway dispatchers and infrastructure managers with valuable insights at the stage of timetable design, such as delay patterns on certain route sections. This allows them to proactively adjust service frequencies and allocate railway resources.

The challenges faced in real-world railway operations highlight that incident occurrence is determined by a complex interplay of factors rather than being reducible to isolated events. For example, extreme weather conditions (Chen and Wang, 2019), cabin crew shortage (Magadagela et al., 2017), overrunning engineering works (Higgins et al., 1999), or the combination of them can lead to varying levels of disruptions, affecting specific stations even the entire service route. In addition, train delays are rarely caused by a single event or disruption and its impacts are not limited within a single station or the service itself. Broadly, delays can be aggravated by a range of factors such as train congestion, platform availability, and signal control. For example, an incident at one station can lead to multiple primary delays on those services that using the same platform or sharing the same route within a certain time window. As such, the relationships between stations and tracks are important to be captured. Traditional approaches, such as mathematical and stochastic models, have been extensively applied to investigate the mechanism underlying delay propagation and predict the occurrence of primary and secondary delays across the railway network. Traditional models may treat stations or lines in isolation, and these delays are investigated at the level of individual stops, which is insufficient for incorporating how event-triggered delays occur/develop across the network. To address this, methods such as graph-based approaches have emerged as valuable tools for understanding delay occurrence and evolution. By embedding stations and their connections, we are able to better understand how incident at one location might initial primary delays, reflecting the real-world, inter-connected nature of railway operations. Such an approach aligns with the broader need to focus on the operational challenges in delay management rather than merely on the computational techniques employed.

Consider a situation like this: a service encountered an incident at its first half journey and such event caused an observed primary delay, however, due to the network-wide effects of its first half part route (for example, speed limitation applied due to traffic demand is very high, and bad weather), the inserted margin time is not sufficient for eliminating the occurred delay. Until the second half part of the route, this primary delay is diminished by a less dense network structure – The primary delay might be eliminated due to the released speed limitation, or the weather condition turns better, or both of them. That is, the complex structure of railway network, involving interconnected tracks, lines, stations, platforms, passenger demand, as well as signaling and other control structures, inevitably have different extent of effects on the occurrence of primary delays. Moreover, a route is not just a simple sequence of stations but a combination of various operational elements such as shared infrastructure, scheduling constraints, and train dynamics (e.g., train dwell times at stations, waiting for other trains at junctions). When a train experiences delay, it often impacts other trains travelling along the same or intersecting routes, a phenomenon known as delay propagation. Traditional methods (e.g., one-hot encoding) that treat each route as an isolated path cannot capture the dynamic interaction between different routes and stations in the network. The route embedding approach compresses the relevant operational data, accounting for the fact that certain stations and tracks are shared across different routes. By embedding these routes, we can capture both the individual station effects and their collective impact on the entire route, addressing the domain-specific need for understanding how delays propagate across shared network elements.

Acquiring meaningful network insights is the first step towards advancing the understanding of service performance. Traditionally, rail transit delays have been addressed at the level of individual stops, but this approach overlooks the network-wide factors contributing to delays, such as bottlenecks caused by shared tracks. By gathering and analyzing such data, it is therefore important, and challenging, for any ML-based method to preserve the corresponding network/route features both globally and locally. For example, paper by Ulak et al. (2020) applied a Bayesian network model and metrics to a study of a track expansion project at the Long Island Rail Road. The results suggest that Bayesian network learning can effectively pinpoint network dependencies and identify rail links and corridors that result in more congestions. From the perspective of operational complexity and network scalability, how to effective incorporate hundreds of stations and multiple routes into delay prediction are one of key challenges in real-world operations. This complexity is amplified in dense urban region where trains run frequently and share infrastructure. From an operations management perspective, railway planners and dispatchers need to predict how a delay at one point in the network will affect others. For instance, a delay in a busy urban station might have ripple effects on multiple lines, potentially leading to bottlenecks or system-wide delays. Traditional methods, such as mathematical models or one-hot encoding, struggle with the network-wide dependencies. By utilizing graph and route embedding, we can significantly reduce the complexity of the data while preserving the essential operational characteristics of the network, leading to more accurate predictions of how delays propagate.

Based on the analysis conducted above, we therefore conclude that comprehending the patterns and structures inherent in various routes is crucial for deriving meaningful insights into primary delays. Within this study, a 'route' refers to the specific path that a service follows from its origin to its destination, and all the intermediate stops along the way. It encompasses the sequence of calling stations. Multiple services may share the same route as long as they call at exactly same origin, destination and intermediate stations. The rationale of our study is: generalized primary delays are inherently relevant to the route information and the characteristics of its passing/serving area. These network characteristics incorporate not only network structure information itself, but also the commonly seen patterns and delay dynamics that occur to the services that running on specific routes. That is, routes with similar characteristics tend to exhibit comparable patterns of primary delays. With this consideration, our goal is to identify an appropriate method for interpreting these hidden patterns cohesively in order to investigate train delay prediction from a novel angle.

We conducted a thorough literature review on predicting delays brought by individual factors. These factors have been fully investigated regarding their impacts on the occurrence of primary delays (see Section 2). Recently, many innovative research directions in train delay prediction are emerging: one promising topic that has been achieved with significant breakthrough is to learn traffic temporal/spatial characteristics before conducting the railway traffic state prediction (Zhang et al., 2021). A number of attempts have demonstrated the feasibility of applying traditional machine learn (ML) algorithms to the procedure of delay prediction in railway systems (Robert and Kim, 2018; Mou et al., 2019). These attempts allow transportation planners and decision-makers to explore the importance of spatial-temporal dependencies in railway traffic planning and prediction tasks. A notable new approach in the study of Heglund et al. (2020) shows that incorporating multiple embedding-based technologies into the process of train delay prediction would be promising in terms of achieving higher accuracy on results.

1.2. *Objectives & research questions*

The ability to predict train delays accurately is essential for railway operators to adjust their rescheduling policies, allocate resources efficiently, and thus improve overall operational performance. From this perspective, our research aims to enhance predictive accuracy and provide insights that can support understanding the delay distribution dynamics and primary delay prediction. Specifically, the goal of primary delay prediction in this study is to predict the generalized primary delay level for a given train service on a given route. The inputs of the proposed model can be summarized as 1) features from a 2-month historical operational train service dataset (including departure/arrival time, margin time inserted for the whole route, speed limit, rolling stock type, and total passenger flow of all the calling stations, etc.) 2) As well as the embedded route characteristics (i.e., connectivity between these stations, travelling time for a given route, and the network density for different sections, etc.) from the proposed SDNE + SVD framework. The outputs of the prediction model is the classification for each instance into a certain category (ranging from delay level 0 to 4 mentioned in section 1.1, represents the degree from no delay to the most severe delays), and then calculate the percentage of correctly classified instances out of the total instances (accuracy). One of the most important features in train delay prediction is the route of a train service, which can be described by a sequence of the stations the train services. However, given a rail network with hundreds of nodes (stations), the representation of a route would imply a large amount of data resource: e.g., if one-hot encoding is used, the route vector's length will be the same as the number of stations. Such a high volume of data requirement will significantly affect the effectiveness or even the feasibility of a supervised learning process.

SDNE (Structural Deep Network Embedding) was firstly developed by Wang et al. (2016) as an effective dimensionality reduction technique in computer science to represent nodes in a graph by short vectors ("embeddings"). Its basic idea is to encourage similar representations for connected nodes (and dissimilar representations for unconnected nodes) in the vector space such that structural relations in the original network can be preserved. Each value in the embedded vector does not have an intuitively explainable meaning but it partially represents the characteristics of a node in a specific dimension. Nonetheless, SDNE itself is unable to solve the problem of route representation since it embeds nodes rather than paths (routes) in a graph.

Based on SDNE, in this paper we propose a novel approach that is able to integrate node embedding vectors of en-route stations (i.e., railway stations that are located along a train service line where trains stop to pick up/drop off passengers during its journey) into a route embedding vector in which more structural information of the target railway network will be compressed and aggregated for reducing the dimensions of route features. The expected route embedding must meet the following criteria.

- Regardless of the length of a specific route, the obtained route embedding vectors must be uniform in size.
- The route representations give the information about overall characteristics of the entire route, such as areas of higher population density and geographic significance, and correspond to the distribution and density of stations along the way (i.e., en-route station clusters).
- Local and global characteristics can be both effectively preserved by route embedding vectors.

With the above background and proposed research objectives, we aim to address the following research questions (RQs).

RQ1: What graph embedding-based method can be applied in the process of encoding structural characteristics and temporal-spatial dependency among network elements? Will the generated node embedding representations effectively preserve the essential information of a network?
RQ2: How can we compute the route embedding vectors by using the node embedding representations?
RQ3: Is possible to improve the prediction accuracy on primary delay by incorporating the route embedding framework with some popular ML-based prediction models, e.g., Decision Tree, Random Forest, and Neural Networks? To what extent will it be boosted?

This paper presents an innovative approach that combines SDNE with Singular Value Decomposition (SVD) to accurately represent train routes and thus to effectively predict train delays. Our methodology makes significant contributions towards the prediction of train primary delays. It incorporates SDNE to capture the intricate relationships and structural properties of the railway network. SDNE offers a powerful means of encoding complex topological information, enabling a more comprehensive understanding of the network's dynamics. By leveraging SDNE, we enhance our model's ability to capture the hidden patterns within the railway network. While the integrated Singular Value Decomposition is a processing pipeline to further enhance the encoding of railway network topology. SVD allows us to reduce dimensionality while retaining essential information, thereby improving the efficiency and effectiveness of our model. This methodological choice not only aids in computational efficiency but also plays a pivotal role in refining the accuracy of our arrival delay predictions. By combining SDNE and SVD into the encoding process, we effectively capture the underlying patterns in the railway network, which in turn, leads to more precise predictions. The major innovative aspects can be stated as.

- We incorporate both network characteristics and historical railway operational data into one framework to perform train delay prediction.
- A deep neural network graph embedding model based on the SDNE algorithm is developed to extract rail network features and generate embedded representation for each station in the network.
- Based on embedded nodes, an SVD-based approach is developed to further represent the route for each train as an important feature for primary delay prediction.
- However, this paper focus solely on predicting 'generalized primary delay', without considering external real-time factors such as weather or signal failures. Our method relies on historical data to predict delays in train design scenarios, e.g., when developing new timetables with modified trains.

As the authors are aware, it is the first attempt using SDNE and SVD-based matrix decomposition technology to represent routes (paths) in a rail network. Our key contribution is for the first time to use SDNE + SVD embedding to represent route information, which is a critical feature in delay prediction. It has the potential to be applied in different train prediction problems as long as route is a feature. We applied our embedding approach on a particular delay prediction problem, i.e., the generalized primary delay prediction to showcase its effectiveness and advantage. The SDNE + SVD method may be applied to a wider scope of problems, which are beyond the scope of the experiments of our paper.

The remainder of the paper is as follows: Section 2 reviews the main approaches that have been proposed in the area of TDP. Section 3 describes the specific graph-embedding/matrix decomposition methods for node and route embedding, respectively, as well as the feature engineering conducted in this study. Section 4 introduces reference network and dataset. Then the process of implementing SDNE into node embedding is demonstrated. Followed with the route embedding generation. Section 5 compares experimental results between our method and its competition, over several benchmarks. Section 6 makes the conclusions and highlights the contributions and limitations of this study. Finally, the future work direction will be identified.

2. Literature review

Multiple factors may generate different extents of primary delays, and the negative impacts of these factors cannot be eliminated and interfered by train operators, which means that the train services will inevitably differ from the scheduled timetables. Numerous studies have been done on railway delay prediction based on various algorithms. Through literature investigation, the existing research can be categorized as follow according to the methods they utilized: Analytical/Regression methods, Graph-based models, Machine Learning models.

2.1. Existing traditional methods

In order to figure out which influencing factors determine the occurrence of train delays and to what extent these factors matters, many studies already being suggested. Traditionally, research methods on this topic can be divided into two categories: statistical distribution analysis (e.g., Goverde et al., 2013; Weng et al., 2014; Yang et al., 2019; Huang et al., 2019; Meester and Muns, 2007; Yuan and Hansen, 2007; Goverde, 2010) and regression models (e.g., Olsson and Haugland, 2004; Gorman, 2009; Ludvigsen and Klæboe, 2014; Tiong et al., 2022). Statistical models have been employed to assess the distributional patterns of railway delays as well as the relationship between delays and their contributing factors. Specifically, track occupancy/release records, as well as other data including information on train movements, are used to build probability distributions for train delays. The calculated distributions of the underlying random variables are subsequently incorporated into various analytical models. These earlier statistical analytical processes show that unanticipated disruptions such as power cable failures, signal system failures or turnout communication disruptions may cause longer operational incident delay, while temporary speed restrictions, overrunning engineering works normally lead to less negative consequences for the subsequent train services. What is certain, nevertheless, is that people are prone to implement regression-based models to discerning delays with various reasons when considering some unanticipated influencing factors (i.e., inclement weather conditions, unexpected passenger occupancy ratios).

Markov models (MM) and Bayesian networks (BN) are another kind of methods for forecasting train delays. We manually categorize them into graph-based method due to these techniques often treat the arrival/departure, running/dwelling of trains as separate events. For example, some studies (Kecman et al., 2015; Şahin, 2017; Gaurav and Srivastava, 2018) considered that the predicted arrival delays were determined by the departure delays at the previous station, while the predicted departure delays were only determined by the arrival delays at the same stations. Markov process only consider the last state to predict the current state in the train operation process. A node (state) can only be linked to another node in Markov models since they view the entire train operation process as a chain (Artan and Şahin, 2023). Similarly, Bayesian networks treat the operation process as a network in which a node may have several links dependent on the logic of the Markov process. The running and dwelling processes were considered edges, whilst arrivals and departures events were considered nodes. China's Wuhan-Guangzhou railway implemented a hybrid BN (Lessan et al., 2019; Huang et al., 2020) for predicting the train delay time, and similar experiments were also conducted by Corman and Kecman (2018) and Huang et al. (2022). Also proposed are generic graph-based models such the active graph (Büker and Seybold, 2012), the temporal event graph (Goverde, 2010; Kecman and Goverde, 2014), and the alternative graph (Corman et al., 2014). With the aim of estimating the precise timing for these events, these graph models took into account the arrivals, departures, passages, and lodging of trains as events. According to Li et al. (2021), the future operating conditions, such as the distance left in the downstream journey and the supplement time distribution, also have an impact on the train operation process.

2.2. *Machine learning-based methods*

More recently, with the technology advances of machine learning applications, a number of studies have demonstrated the feasibility of applying ML algorithms to collect, process, and analyze large amount of data and extract useful information for TDP problems (e.g., Nilsson and Henning, 2018; Peters et al., 2005; Mou et al., 2019; Marković et al., 2015; Li et al., 2016). These models develop broad rules/non-linear correlations that links inputs to desired outputs via the iterative model training. Compared to statistical and regression models, they often offer more effective fit but less straightforward interpretation. For example, Nilsson and Henning (2018) applied the classical decision tree classifier with and without AdaBoost to compare the different hyper-parameter settings in predicting train delays. In addition, although the neural network technology was developed in the late 20th century, it is becoming more widely used as a method for delay prediction, with its multiple variants techniques. For instance, according to Peters et al. (2005), the multi-layer neural networks have the capability of simulating the biological archetypes of the real-world train network in an effective manner, such that the prediction of a particular train delay is determined by the principle of classical pattern matching: the network must be trained so that various delay scenarios - represented by a specific assignment of the input neurons - will result in corresponding delay patterns at the output neurons. As a new advanced branch of artificial neural network, a long short-term memory based predictor was constructed by Mou et al. (2019) for illustrating the interaction between the factors affecting train delay. Furthermore (Huang et al., 2021), introduces FCF-Net, a deep learning model combining fully connected neural networks (FCNN) and convolutional neural networks (CNN) for efficient and accurate recognition of train delay propagation patterns in railway systems, demonstrating superior performance compared to conventional deep learning and other state-of-the-art models when tested on data from two high-speed railway lines in China.

In fact, ML-based methods have been such intensively introduced in literature to address train delays prediction that we are not able to identify which model/algorithm is better over others for predicting train delays more accurate. Marković et al. (2015) utilized support vector machine (SVM) to analyze train arrival delays on Serbian Railways and it shows a better prediction accuracy compared with the outputs of artificial neural network (ANN) model. However, comparison of the results provided in the study of Li et al. (2021) indicates that the random forest outperformed the other evaluated methods (i.e., gradient-boosted decision tree (GBDT), extreme gradient boosting model (XGBOOST), and ANN). A data-driven Train Delay Prediction System (TDPS) for large-scale railway networks, leveraging contemporary big data technologies, specifically employing a fast learning algorithm for Shallow and Deep Extreme Learning Machines, has been proposed by Oneto et al. (2018) on Italian railway network, demonstrates significant improvement in predicting train delays. Another trend through the literature we discovered is that the hybrid method that assembles a set of simulation and statistical approaches as good attempts have been explored. For example, a k-nearest neighbor (KNN) model was employed by Li et al. (2016) where the linear regression model was used to estimate the peal hour dwell times while KNN model was specifically for predicting the length of dwell time within off-peak hours. Similarly, Nair et al. (2019) introduced a simulation-based model which combined random forest and kernel regression for forecasting train delays in Germany passenger service network. The obtained performance illustrate that this ensemble method outperformed its component models.

2.3. *Graph embedding approaches*

The structure of railway networks is complex, involving interconnected tracks, lines, stations, platforms, as well as signaling and other control utilities. It is therefore important, and challenging, to preserve the corresponding network features both globally and locally. Most of the existing studies deal with this by using Autoencoders (AEs). AEs are unsupervised artificial neural networks designed for data generation through encoding and decoding processes. The main goal of the AE is to copy its input as the desired output, and to automatically capture the most relevant and representative features from the input data. Also, AEs are employed to get a latent space smaller than the original data dimension. Some applications of AEs such as Principle Component Analysis (PCA) (Abdi and

Williams, 2010), Linear Discriminant Analysis (LDA) (Xanthopoulos et al., 2013), and Discriminant Function Analysis (DFA) (Büyüköztürk and Çokluk-Bökeoğlu, 2008) have been implemented in data analysis for various sectors. The main difference between AEs and graph embedding techniques is that AE represent input using non-linear combinations of derived features without any supervised component, while graph embedding approaches are typically semi-supervised or supervised, which allows a decoder to fine-tune encoding parameters for updating the generated representations iteratively. For example, by analyzing the large-scale traffic congestion data with a deep autoencoder (Zhang et al., 2019), successfully learnt temporal correlations of a transportation network and predicting traffic congestions.

On the other hand, according to Cui et al. (2018), embedding is a powerful machine learning tool that is commonly used for transforming and interpreting complex input objects such as texts, images, and graphs into a low-dimensional vector space. Such technique essentially construct a similarity graph for a set of D-dimensional nodes (D is the number of available features of nodes) based on their neighborhood information and then embed each node of the graph into a d-dimensional vector space, where $d \ll D$, and such mapping function we called as "Encoder" (*ENC*). Fig. 1 illustrates the embedding as the transformation of a discrete object into a vector of continuous numbers: embedding convert each individual station to a set of numerical representations in Fig. 1a (various colors of cells in the output represents different values of digits). Here is an example of the graph embedding demonstration. For illustrating how we process each of the station to a set of numerical representations (vectors). Another example given is the embedding process of encoding a network into a d-dimensional space (Fig. 1b). Graph typology, vertex-to-vertex relationship, and other relevant information about graphs would be captured by graph embedding algorithm in the context of graph embedding technique family, such that each node in the original network will be projected into the embedding space accordingly (Hamilton et al., 2017).

The first step in performing graph embedding is to embed nodes. Some effective methods has been proposed in previous literature. They can be broadly classified in two categories. The first, graph factorization method, is inspired by the classic matrix factorization techniques, and aims to find latent/unobserved factors associated with nodes which can be used for further inference for dimensionality reduction and multi-dimensional scaling (Plummer, 2007; Ahmed et al., 2013; Belkin and Niyogi, 2001; Cao et al., 2015). The second, random walk approach, is to optimize node embedding by employing a stochastic measure to compute node similarities, instead of using a deterministic measure in factorization-based methods (Grover and Leskovec, 2016; Perozzi et al., 2014). The majority of these node embedding algorithms are derived from what we call "shallow node embedding". Where the encoder function mapping nodes to vectors embedding is simply an "embedding lookup table" rather than generating embedding based on the information of neighboring nodes.

From the perspective of a broader transport domain (i.e., road transportation), a further review of state-of-the-art applications and extensions against graph embedding approaches has been conducted. A novel neural network developed by (Shen et al., 2020), aims for addressing the challenge of accurate trajectory travel time prediction. It utilizes tensor decomposition and graph embedding to effectively extract travel speed and road network structure information from historical trajectories. Empirical results from two real-world datasets demonstrate that the graph embedding method outperforms its counterparts, offering significantly improved performance and robustness in travel time prediction. Similarly (Kang et al., 2019), identified the massive traffic flow data generated by modern transport systems presents a significant challenge in accurately capturing both the hidden spatial patterns and the temporal dynamics from these data. They therefore introduced a novel model called spatial-temporal graph self-attention (STGSA) for short-term traffic flow prediction. This model effectively captures network-level spatial relationships through graph self-attention layers while also handling temporal embedding leveraging the integration of RNN cells with Gated Recurrent Units. The recent literature has emphasized the significance of graph embedding techniques in addressing the tasks of accurate trajectory-based travel time prediction and traffic flow analysis. Emphasizing its ability to handle both network-level spatial and temporal dependencies. These studies collectively underscore the growing importance of graph embedding approaches in advancing traffic prediction tasks in general transportation sectors.

2.4. Summary and research gap

The existing solutions to the TDP problem can be categorized into several groups: attributing observed historical delays based on various influencing factors (distribution estimation), formulating the delay dynamics with certain mathematical linear functions/distributions (probability distribution fitting), incorporating historical data into the analytical process of regression (correlation

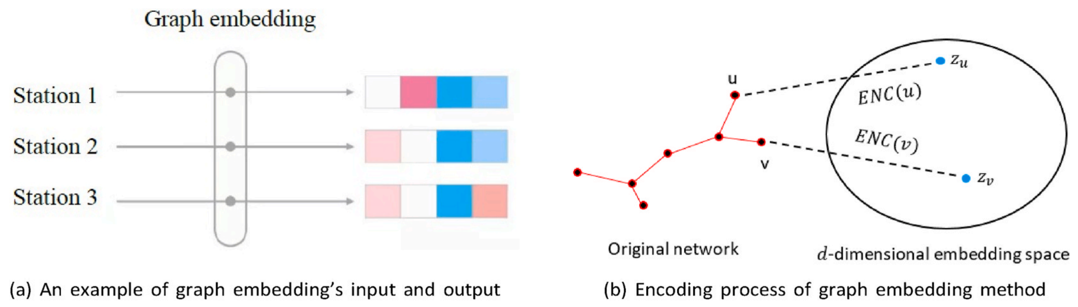

Graph embedding
Station 1
Station 2
Station 3
(a) An example of graph embedding's input and output
u
v
$ENC(u)$
$ENC(v)$
z_u
z_v
Original network
d-dimensional embedding space
(b) Encoding process of graph embedding method

Fig. 1. Graph embedding transformations.

analysis), simulating the temporal-related events and their propagation (graph-based methods) for real-time applications such as interactive timetable and decision support system to assist train dispatchers, as well as various ML-based methods (see Table 1). There is no definitive evidence that some specific algorithms always outperform others. The optimal solutions vary due to differences in the train operating environment, amount of available data acquired, objectives of the study, performed pre-processing strategies, etc. Therefore, algorithm selection as an effective approach to improve the performance of output, is necessary, and the algorithm with the best accuracy should be chosen to establish the delay prediction model.

In addition, we reviewed the data types utilized in previous delay predictions, and categorized them into five classes: timetable data, route information, network topology structure, infrastructure data, and profile of operating train (shown as the last five columns in Table 1). Data that can be estimated based on actual and projected timetables is referred to "timetable data" (i.e., arrival/departure delays, margin times, and headways). The route information comprises of the specific edges that trains will pass through when traversing in the network, including the sequence of stations they will use. Network topology structure refers to the geographical locations of all stations, the average travelling time on the edges, and the connectivity status between each pair of stations. Data of infrastructure, such as the length of various sections, can reflect the infrastructure/facility resources on stations. Train operating profile includes information such as, type of locomotive, speed limitation, passenger occupancy ratio, etc.

From the tactical level, most of the existing works in predicting primary delay failed to consider the important role of 'route', as

Table 1
A summary of the review of delay prediction studies: proposed methods and utilized data.

Model type	Method	Literature	Timetable data	Route information	Network topology structure	Infrastructure data	Operating train profile
A&R	PD	Goverde et al. (2013) Weng et al. (2014) Yang et al. (2019) Huang et al. (2019)	✓				
	DE	Meester and Muns (2007) Yuan and Hansen (2007)	✓			✓	
	CA	Olsson and Haugland (2004) Gorman (2009) Ludvigsen and Klæboe (2014)		✓		✓	✓
GB	MC	Kecman et al. (2015) Şahin (2017) Gaurav and Srivastava (2018)	✓				
	BN	Corman and Kecman (2018) Lessan et al. (2019) Huang et al. (2020)	✓				
	ACG	Büker and Seybold (2012)	✓				
	ALG	Corman et al. (2014)	✓			✓	
	TEG	Goverde (2010) Kecman and Goverde (2014)	✓	✓			✓
ML	DT	Nilsson and Henning (2018) Wang and Zhang (2019)	✓	✓			✓
	NN	Peters et al. (2005) Yaghini et al. (2013)		✓		✓	✓
	LSTM	Mou et al. (2019)	✓				✓
	SVM	Marković et al. (2015) Barbour et al. (2018)	✓	✓		✓	✓
	RF	Li et al. (2021)	✓			✓	✓
EM	KNN; LR	Li et al. (2016)	✓	✓			✓
	RF; KR	Nair et al. (2019)		✓		✓	✓
Graph embedding; Matrix decomposition	SDNE; SVD DT; RF; MLP	This paper	✓	✓	✓	✓	✓

The table categorizes delay prediction models into four types: **A&R** (Analytical and Regression models), **GB** (Graph-Based models), **ML** (Machine Learning models), and **EM** (Ensemble models). The 'Method' column uses abbreviations for compactness: **PD** (Probability Distribution fitting), **DE** (Distribution Estimation), **CA** (Correlation Analysis), **MC** (Markov Chain), **BN** (Bayesian Networks), **ACG** (Activity Graph), **ALG** (Alternative Graph), **TEG** (Time-Event Graph), **DT** (Decision Tree), **NN** (Neural Networks), **LSTM** (Long-Short Term Memory), **SVM** (Support Vector Machine), **RF** (Random Forest), **KNN** (K-Nearest Neighbors), **LR** (Linear Regression), **KR** (Kernel Regression), and **GE** (Graph Embedding). Columns represent types of data utilized in studies: **Timetable Data**, **Route Information**, **Network Topology Structure**, **Infrastructure Data**, and **Operating Train Profile**.

they are difficult to be represented as a feature that can be effectively used by a ML model. However, naïve methods such as one-hot encoding may be insufficient. This limitation is evident when comparing analytical and regression models with more versatile machine learning models that are capable of integrating a diverse range of data types for training. Research Question 1 (RQ1) of this study emerges as a pivotal point for bridging this gap, as it seeks to understand the capability of graph embedding-based methods in encoding the route structural characteristics and spatial dependencies among route elements. This question aims to explore whether such methods can effectively preserve essential route information, ultimately addressing the need from a route-based perspective in train delay prediction. Furthermore, at the technical level, there is a noticeable absence of evidence in the literature regarding the aggregation of node embedding into meaningful route embedding in the context of train delay prediction. Which represents a significant gap in the current research landscape. Our paper takes this opportunity as an innovation point and provides an in-depth exploration. Research Question 2 (RQ2) aligns with this innovation and aims to investigate the methodology for computing route embedding vectors using node embedding representations. When it comes to how to effectively apply graph embedding methods to practical railway delay prediction scenarios, RQ2 has been proposed and it serves as such a technical bridge. It fills the gap from meaningless and difficult-to-understand graph embedding representations into a data structure that is more condensed, highly relevant to a specific service route, and can be directly used by machine learning models.

Lastly, existing literature exploring the feasibility of applying graph embedding to delay forecasting is scarce. Although a few works have considered spatial dependence and temporal correlations, there has been no attempt to extract the network topology structure using graph embedding, not to mention to combine them with classical ML predictors for implementation. This research gap is an essential motivation for our study. Research Question 3 (RQ3) directly responded such concern, seeking to enhance prediction accuracy on primary delays by integrating the route embedding framework, obtained through graph embedding, with popular machine learning classification models. This RQ, therefore, aims to quantify the extent to which such an integrated approach can boost the accuracy of train delay prediction, filling a critical gap in the current state of train delay prediction research.

Notably, the foundation of our study rests on a supposition that the manifestation of ‘generalized primary delays’ is intricately interconnected with the specific route traversed by a train and the distinct characteristics inherent to the geographic areas encompassed by that route. These encompassing features represent not only the intricate network structure, but also encompass the implicit interplay of common events and recurrent patterns unique to each section/or the whole of the route. For example, external factors such as weather are not considered explicitly in our prediction model but indeed they are implicitly included in the model (i.e., if a route often experiences bad weather and thus frequent delays in the past record, the model will be able to reflect this accordingly). Our research endeavors to advance the comprehensive understanding of this intricate relationship by introducing a suitable framework. This framework is designed to transcend conventional depictions of network topology dependencies by encapsulating the compound attributes that collectively shape the performance of a train service in terms of ‘generalized primary delay’. These attributes are derived from an extensive analysis of historical train service data that has traversed the same or closely related routes. Consequently, such framework is able to offer an enhanced capacity to model and predict the hidden interdependencies governing train service performance and its ultimate impact on generalized primary delay.

3. Methodologies

In this study, we will take one-hot encoding for routes, as well as PCA method as benchmarks to assess the capability of our proposed model and make comparisons accordingly. These benchmarks were chosen because they represent commonly used techniques for dimensionality reduction and feature representation, but with its own limitations in capturing network interdependencies as a whole. A comprehensive introduction about the utilized one-hot encoding and implemented PCA algorithms are included in Appendix. Furthermore, the predictive performance on the given dataset over these benchmarks and our proposed method will be compared and discussed in Section 6, highlighting their relevance to the studied problem.

3.1. A graph-embedding method for describing a railway network

The most essential part for our methodology framework is the acquisition of structural deep network representation. We proposed a model based on SDNE method to represent stations in the railway network. This model is inspired by the recent successful implementation of deep learning techniques that firstly derived from Wang et al. (2016), and it has since been demonstrated by others for its powerful embedding capability on audio speech, image grids and texts (Bengio et al., 2009; Hinton et al., 2012; Krizhevsky et al., 2012; Socher et al., 2013).

The relevance of using SDNE to describe stations in the railway network lies in its ability to represent the spatial and temporal relationships among stations. Specifically, SDNE allows us to embed the topological structure of the network in a way that reflects real-world operational dynamics. SDNE captures first-order proximity, representing direct connections between stations (e.g., sequential stops along a route), and second-order proximity, which incorporates the influence of shared neighboring nodes (e.g., interlinked routes, shared platforms, or cascading delays across regions). These proximities are not merely abstract metrics—they correspond to tangible operational factors such as platform sharing, train congestion, and signaling dependencies that influence delay occurrence.

Additionally, the temporal dimension of delays—how an incident at one station evolves over time—is inherently intertwined with the spatial relationships among stations. SDNE encodes this temporal-spatial interaction by preserving the structure of the network while training embedding vectors that prioritize delay-relevant relationships. For example, a disruption at a central hub station might cause ripple effects across multiple branches of the network, affecting downstream connections over time. The SDNE framework, through its joint optimization of encoder-decoder layers and proximity-aware loss functions, ensures that such cascading effects are accurately represented in the low-dimensional embedding space. This ability to encode both immediate and network-wide dependencies makes SDNE particularly suited for railway delay prediction.

Fig. 2 illustrates our proposed SDNE framework for embedding a railway network. This framework uses the original railway network characteristics as input for the encoder-decoder layers, whose detailed structure includes the defining of first-order and second-order proximity and the identification of connectivity status between any two nodes. In order to update the resulting embedding vector to satisfy the lowest overall loss costs during training, loss functions corresponding to each proximity in the output layer will provide the encoder-decoder with the optimized parameters. As a result of this, each node in the network is given its final low-dimensional embedding representation.

According to the Encoder-Decoder framework shown in Fig. 2 below, a set of terms and notations have been defined within Table 2. Note that symbol above the relevant parameters denotes the parameters of the decoder.

The original railway network is typically represented as a graph $G(V,E)$, where $V = \{v_1,\ v_2, \dots,\ v_n\}$ is the set of railway stations in the graph, and $E = \{e_1,\ e_2, \dots,\ e_p\}$ is edge set composed of immediately successive station nodes in the geographical space. That is, if there is an observed edge between station node 1 and station node 2, $e_1 = A_{1,2} = 1$ (the first element $A_{1,1}$ shows the connectivity status of node 1 with itself). Therefore, we can obtain $(n \times n)$ adjacency values regarding all station nodes $A_{n,n} = e_p \cup 0$, which contains n instances $A_1, A_2, \dots, A_n$. For each elements $A_{u,v} = 1$ in the instance A_n, if and only if there exists an edge between node u, v. i.e., A_n provides the information of the neighborhood structure of node v_n.

The encoder compresses the input and the decoder attempts to recreate the input from the compressed version provided by the encoder. The definition of the deep auto encoder-decoder has been given in equations (1)–(3) below. Given the input X, the hidden representation for each layer are shown as follow:

$$y_i^{(1)} = \sigma\left(W^{(1)} x_i + b^{(1)}\right) \tag{1}$$

$$y_i^{(2)} = \sigma\left(W^{(2)} y_i^{(1)} + b^{(2)}\right) \tag{2}$$

$$y_i^{(k)} = \sigma\left(W^{(k)} y_i^{(k-1)} + b^{(k)}\right) \tag{3}$$

The encoder has multiple non-linear functions that map each input data vector $x_i (i = 1, 2, \dots, n)$ in data matrix X to the representation space and the decoder also contains some non-linear functions mapping the representation in embedding space to reconstruction space. It is worth to mention that we define the sigmoid function σ in our model as equation (4):

$$\sigma\left(W^{(k)} y_i^{(k-1)} + b^{(k)}\right) = \frac{1}{1 + \exp\left(-W^{(k)} y_i^{(k-1)} - b^{(k)}\right)} \tag{4}$$

Each obtained $y_i^{(k)}$ is the raw node-embedding vector for node i $(i = 1, 2, \dots, n)$. However, the values of $y_i^{(k)}$ here are a set of semi-outputs that have not been updated by the loss function. In order to obtain the final-state meaningful node embedding vectors, we can then obtain the output $\widehat{x}_i$ by reversing the calculation process of the encoder. Loss function is imposed of this framework to minimize the reconstruction loss such that the node representations can be updated with a greater preservation on proximity by the reversing procedure. The goal of the loss function we built is to minimize the reconstruction error of the output $\widehat{x}_i$ and the input x_i. The basic form of our proposed loss function can be defined as equation (5):

$$L = \sum_{i=1}^{n} \|\widehat{x}_i - x_i\|_2^2 \tag{5}$$

One of the expected outputs of graph embedding is to preserve both local and global structure effectively. In our approach, we compile the connections between any two stations: whether they are directly connected, share the same neighboring station, or nothing at all. For the first situation and two situations in the latter, we further apply the first-order and second-order proximity method, respectively, as introduced by Tang et al. (2015).

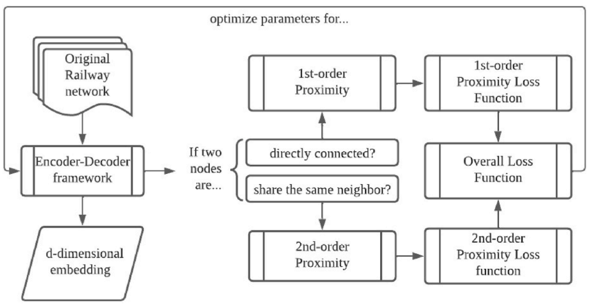


Fig. 2. High-Level architecture of SDNE model.

Table 2
Notations and Explanations of SDNE model.

Notation	Explanation
n	Number of nodes in the network
p	Number of edges in the network
V	Set of nodes in the network, where $V = \{v_1,\ v_2, \ldots,\ v_n\}$
E	Set of edges in the network, where $E = \{e_1,\ e_2, \ldots,\ e_p\}$
K	Number of layers in the model, where $k\ = \{1, 2, \ldots, K\}$
$A_{u,v}$	The adjacency matrix for the network, where $u, v \in V$
x_d	The input data vector, where $d\ = \{1, 2, \ldots, n\}$
$b^{(k)}$	The k-th layer biases in encoder
$\widehat{b}^{(k)}$	The k-th layer biases in decoder
$\widehat{x}_d$	The reconstructed data vector, where $d\ = \{1, 2, \ldots, n\}$
X	The input data matrix, where $X = \{x_1,\ x_2, \ldots,\ x_n\}$
$\widehat{X}$	The reconstructed data matrix, where $\widehat{X}\ = \{\widehat{x}_1, \widehat{x}_2, \ldots, \widehat{x}_n\}$
$Y^{(k)}$	The k-th layer hidden representation, where $Y^{(k)}\ = \{y_1^{(k)}, y_2^{(k)}, \ldots, y_n^{(k)}\ \}$
$W^{(k)}$	The k-th layer weight matrix in encoder
$\widehat{W}^{(k)}$	The k-th layer weight matrix in decoder
L	Loss function
B	Set of biases in encoder, where $B\ = \left\{b^{(1)}, b^{(2)}, \ldots, b^{(k)}\ \right\}$
σ	The sigmoid function
α	Parameter for L1-norm regularization term
γ	Parameter for L0-norm regularization term
θ	The collection of parameters $\left\{W^{(k)}, \widehat{W}^{(k)}, b^{(k)}, \widehat{b}^{(k)}\right\}$

The first-order proximity describes the pairwise proximity between vertices/stations: if an edge is observed between a pair of stations u and v, a weight corresponding to that edge is defined as the average travel time for all observed trains running between station u and station v, indicating the first-order proximity between u and v. Otherwise, their first-order proximity is 0. The second-order proximity compares the neighborhood network structure between two vertices: if they share multiple common neighbors, they tend to be very similar even they are not directly connected to each other.

We can observe some edges in the network, but many legitimate connections between nodes are not visible, which means that edges between vertices do imply their similarity however a lack of edges does not always indicate dissimilarity. In railway network, the number of non-zero elements $A_{u,v}$ are far less than that of zero elements. Thus we impose more penalty to the reconstruction loss of the non-zero elements than zero elements. We combine the loss function of first-order ($L_{1st-order}$) and second-order proximity ($L_{2nd-order}$), and jointly minimize them by the following objective function (6):

$$L_{mix} = L_{2nd-order} + \alpha L_{1st-order} + \gamma L_0 = \|(\widehat{X} - X)^{*}B\|_F^2 + \alpha \sum_{u,v\in V}^{n} A_{u,v}\left\|Y_u^{(k)} - Y_v^{(k)}\right\|_F^2 + \gamma L_0 \tag{6}$$

Where L_0 in equation (6) is a L2-norm regularization parameter used here for preventing overfitting, the definition of it can be given as below:

$$L_0 = \sum_{k=1}^{K}\left(\|W^{(k)}\|_F^2 + \|\widehat{W}^{(k)}\|_F^2\right) \tag{7}$$

During the layer-to-layer training (i.e., k-layer encoding and decoding), α and γ are two parameters we want to optimize, such that L_{mix} can be minimized. Finally the objective function (mixed loss function) of the proposed model can be written as:

$$L_{mix} = \|(\widehat{X} - X)B\|_F^2 + \alpha \sum_{u,v\in V}^{n} A_{u,v}\|Y_u^{(k)} - Y_v^{(k)}\|_F^2 + \gamma \sum_{k=1}^{K}\left(\|W^{(k)}\|_F^2 + \|\widehat{W}^{(k)}\|_F^2\right) \tag{8}$$

3.2. Singular Value Decomposition for route embedding

The SDNE model is only able to generate a node embedding vector for each station. A key issue to be further considered is to do route embedding - since this can be used to represent the route feature of each train, which is highly important in characterizing the train services. Simply concatenating the node vectors of the stations a train travels through is not workable as the lengths of routes can vary significantly and it will also give a tremendous long vectors to represent routes like one-hot encoding. We thus need to model the network topology specifically from the route perspective. To this aim, Singular Value Decomposition (SVD) is introduced for generating route vectors. SVD is a powerful tool for extracting matrix information (Yang et al., 2011). A detailed description about how to implement SVD method has been illustrated in the part of 'Introduction of Singular Value Decomposition' in Appendix. To the best of author's knowledge, it is the first time to introduce such matrix decomposition technology into train delay prediction procedure.

In this study, we will perform this SVD strategy on node embedding vectors and use the obtained singular value matrix to represent the route embedding vectors, such that the route length of different train services can be normalized, and the dimensionality of original network features can be reduced significantly without missing any essential topology structure information.

With the aim of predicting delay level for each service, we extract the route information (i.e., sequences of en-route stations) from the original station matrix in this step, with the matrix decomposition technique. There are several questions we need to consider before using SVD to generate route embedding vectors from node embedding: 1) The acquired route embedding vectors need to be uniformed on their size regardless of the length of a particular route. 2) The properties of the entire route, such as if the stations locate more condensed or sparse on specific sections than others, station sequences, and the level of congestion along the route, can be effectively retained in the route representations. 3) The dimensions of generated representation should be significantly reduced compared to the original one. In order to satisfy all of these requirements, we propose the route embedding framework shown in Fig. 3. Assume that there are S stations in the rail network and they are embedded into vectors of length l by our SDNE model. Let $t \in T$ be the set of routes, $N \in \mathbb{R}^{S\times l}$ be the matrix storing all node embeddings and let r be the length of our resultant route embedding vectors to be found. Let $\rho(t)$ be the resultant route vector after embedding for route t. Our route embedding algorithm is summarized by Algorithm 1 shown in Table 3.

There are many heuristic strategies for determining the number of singular values to keep – it is important to ensure that the selected dimension of singular values can avoid any redundant features but also include the necessary information in as much as possible. In practice, studies that discuss how many percentage of the entropy/essential information should be retained from the original matrix. For example, Rissanen (1978) and Tanwar et al. (2018) had researched and investigated for deciding how many percentage of entropy information should be kept from $\Sigma(t)$. However, the actual percentage to retain can vary depending on the specific problem and desire level of accuracy. By considering the past experimental settings in (Banerjee and Pal, 2014), the choice of $\eta = 90\%$ in this case has been referred, which comes from the notion that the first few singular values typically capture the majority of the variation in the original matrix, and the remaining principal components contribute relatively little. In this study, we assess different digits if the chosen singular values can effectively preserve over 90% necessary information of the original matrix. The result shows when we keep three dimensions of singular values the preserved entropy can reach 93% however when we keep this number as 2 it does not.

Fig. 4 illustrates a comprehensive diagram that delineates the proposed methodology framework as a whole, spanning from the incorporation of utilized datasets to the subsequent model evaluation stage. This graphical representation encapsulates distinct modules, namely the Data pre-processing module, SDNE + SVD module, and Delay prediction module, along with elucidating the data flow and interactions inherent to these components. The input datasets encompass the Adjacency matrix and 2-month historical operational data. Within the SDNE framework, input parameters include link travel time, distances, connectivity of stations, and route information of services. In contrast, the comparative method, PCA, exclusively processes raw Route information. The Data pre-processing module involves the application of various data cleaning and pre-processing strategies according to specific data types. A more detailed exposition of these strategies will be provided in next section. The Delay prediction module merges outputs from the aforementioned modules, concatenating them into a singular, comprehensive numerical vector. This resultant vector serves as input for Machine Learning classifiers during the training phase. Ultimately, predictions are obtained from three traditional machine learning-based predictors. Performance evaluation of the proposed method involves a comparison between predicted labels for each service and the corresponding ground truth labels. This comparative analysis allows for the validation and assessment of the method's effectiveness.

4. Computational experiments

4.1. Dataset description & pre-processing

We applied the above described graph-embedding framework to the TransPennine Express (TPE)[2] rail network. The data provider TransPennine Express, is a train operating company in Britain, which provides rail services in Northern England and Scotland. There are 3 route groups/corridors (shown as red, light blue, and purple routes in Fig. 5, respectively) connecting the Great Manchester area and cities such as Glasgow, Liverpool, Leeds, and Newcastle. However, there are more train services/delay instances utilized for modelling. Each train service corresponds to a route composed of a set of sequential stations. Different services may share the same route. The routes would be different if the serving stations between two services are not exact same, even though they may share the same route group/corridor. Train timetable and running delays between 28/05/2017-24/06/2017, as well as 27/05/2018- June 23, 2018 was used in our experiments. Notably, the TPE provided dataset does not include all train services operated/delivered during the considered 2-month period - only a small fraction of them were provided. However, even the number of included delays is relatively small, one of contributions of this research is to provide an innovative tool for prediction based on limited data sources, which holds the promising potential to be applied in a large-sample case. Specifically, the raw dataset was represented as a set of individual timetables. Each timetable/train service is identified with a unique Train ID. The Dates related to these timetables specified as a period rather than a certain date (e.g., May 19, 2018 to December 10, 2018). That is, each primary delay record in our data is still associated with a specific date but it is not given in the data and only the whole period is given. A given record shown with primary delay indeed

2 https://www.tpexpress.co.uk/.

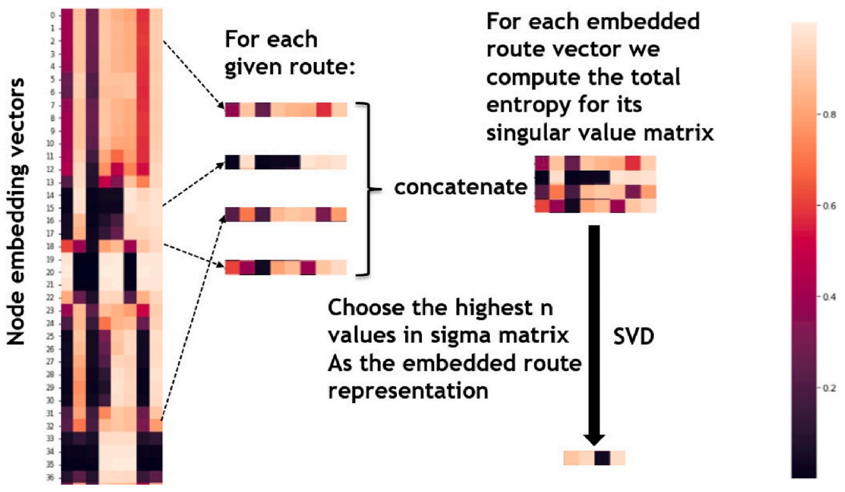

Node embedding vectors
For each given route:
concatenate
For each embedded route vector we compute the total entropy for its singular value matrix
SVD
Choose the highest n values in sigma matrix As the embedded route representation
0.8
0.6
0.4
0.2

Fig. 3. Process of performing route embedding.

Table 3
Algorithm 1: Calculating route embedding vector $\rho(t)$ for each $t \in$

Step 1: Acquire $\Sigma(t)$
 For all $t \in T$ **do**

1) For each route t in the timetable with K stops, we retrieve the node vector of its origin, intermediate and terminal stations $v_1(t), v_2(t), \ldots, v_K(t) \in \mathbb{R}^{1\times l}$ from the node matrix N with their correct sequence on the route.
2) Let matrix $R(t) = \begin{pmatrix} v_1(t) \\ v_2(t) \\ \vdots \\ v_K(t) \end{pmatrix} \in \mathbb{R}^{K\times l}$ be a concatenation of the node vectors.
3) Apply SVD to $R(t)$ to get $\Sigma(t)$, i.e., $R(t) = U(t)\Sigma(t)V(t)^T$.

Step 2: Define *keep_info_num*$(R(t),\eta)$
Define *keep_info_num*$(R(t),\eta)$, $t \in T$ as a function that returns the minimal number of elements that should be kept in $\Sigma(t)$ after applying SVD to $R(t)$ such that at least a percentage η of the information of $R(t)$ can be retained in $\Sigma(t)$. Let the $\Sigma(t)[k]$ be the k-th element in $\Sigma(t)$:

$$keep\_info\_num(R(t),\eta) = \min\left\{ k : \begin{pmatrix} \Sigma(t)[1] \\ \Sigma(t)[2] \\ \vdots \\ \Sigma(t)[k] \end{pmatrix}^2 > \Sigma(t)^2 \cdot \eta \right\}$$

Step 3: Calculate r
Let each v_i in $N(i = 1, 2, \ldots, S)$ **be** $R(t)$ in **Step 1**
$r_{min} = \max_{i=1,2,\ldots,S}\{keep\_info\_num(v_i,\eta)\}$; note that v_i is a theoretically shortest route that only contains one node.
Let N **be** $R(t)$ in **Step 1**

$$r_{max} = keep\_info\_num\left(\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_S \end{pmatrix}, \eta\right);\text{ note that } \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_S \end{pmatrix} \text{ is the theoretically longest route that contains all nodes.}$$

Then we calculate $r = \max(r_{min}, r_{max})$.
Step 4: Determine $\rho(t)$
$\rho\rho(t) = diag(r, \Sigma(t))$, where $diag()$ means to extract the most significant r elements from the diagonal $\Sigma(t)$.

represents a certain service running during this period but unfortunately full details regarding what specific time or date that record was collected were unavailable to the authors. The cleaned dataset consists of 1191 train delay instances in the network of 177 stations and 192 edges. Table 4 gives the information about how many services were used for prediction on each route group.

Train operations data was sourced from the TRUST [Trains Running Under System TOPS (Total Operation Processing System)],[3] a Network Rail computer system that monitors the progress of trains and tracking delays on Great Britain's rail network - It records the specific location of trains on the network continuously by the signaling system. This is undertaken using track detection - on the GB network this is predominantly track circuits and axle counters. In some more remote areas, signalers record actual running times manually. TRUST Delay Attributor (DA) is used for delay attribution and uses a feed from TRUST. When there is a delay the TRUST DA

[3] https://en.wikipedia.org/wiki/TRUST.

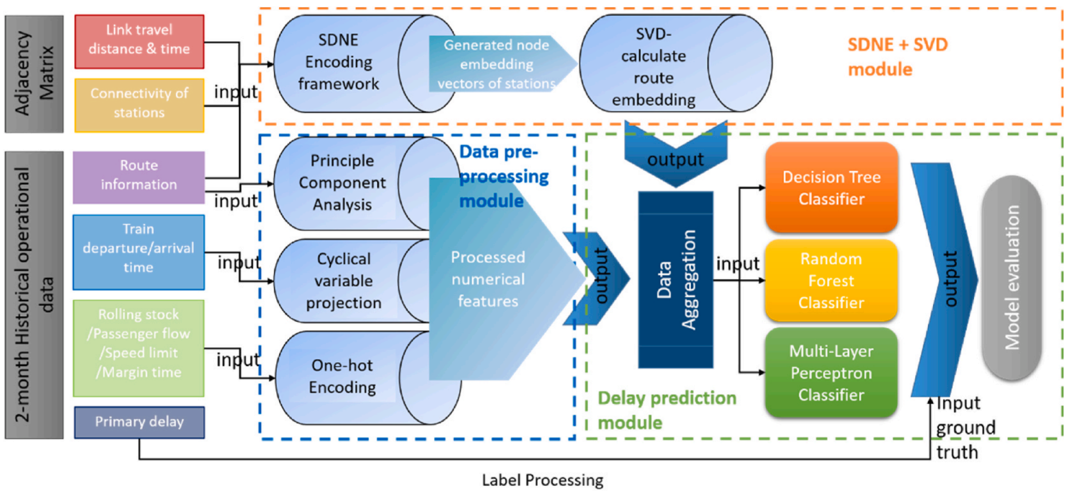

Adjacency Matrix
Link travel distance & time
Connectivity of stations
2-month Historical operational data
Route information
Train departure/arrival time
Rolling stock /Passenger flow /Speed limit /Margin time
Primary delay
input
SDNE Encoding framework
Generated node embedding vectors of stations
SVD-calculate route embedding
SDNE + SVD module
Principle Component Analysis
Cyclical variable projection
One-hot Encoding
Data pre-processing module
Processed numerical features
output
Data Aggregation
Decision Tree Classifier
Random Forest Classifier
Multi-Layer Perceptron Classifier
Delay prediction module
Model evaluation
Input ground truth
Label Processing

Fig. 4. All-encompassing chart of proposed methodology.

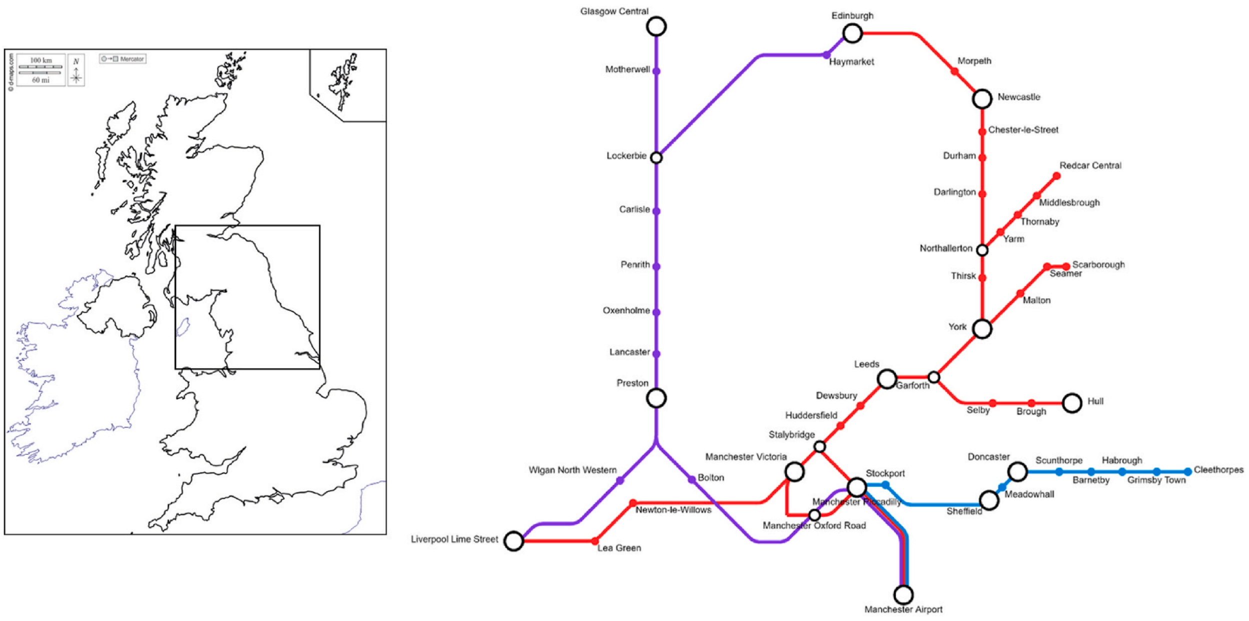


Fig. 5. The TPE network and its three route groups (represented by red, purple and blue lines). (For interpretation of the references to color in this figure legend, the reader is referred to the Web version of this article.)

Table 4
Number of services per route group.

Route Group	Number of services	Number of distinct routes
Red route group	917	148
Purple route group	99	33
Blue route group	175	32

system identifies where that delay has occurred and creates an alert allowing users to investigate the cause of delay and attribute it accordingly. When a delay alert is created in TRUST DA, it appears on the screen of a Level 1 Train Delay Attributor (TDA) at the relevant Network Rail route. The TDA then begins the process of investigating and attributing it. The TDA will undertake a preliminary investigation to determine whether the alert is a new delay or is linked to an existing delay. In this study we were only provided with those delays categorized as new delays (primary delays), so our research mainly focus on these delays that do not linked to another delay. That is, the data provided to authors do not contains any secondary delay-related cases.

Table 5 gives information about those features used for modelling in detail and gives the explanation for each of them. Inputs of the proposed framework include the normalized variables from ‘temporal features’, ‘numerical features’, ‘categorical features’, and ‘network features’ stated in Table 5. In addition, we give the information about how the dimensionality of each feature changes with the implementation of feature engineering or dimensionality reduction strategies. The dimensions describe 1) the raw dataset obtained from TPE, 2)performing PCA on network features and one-hot encoding/normalization on other features, and 3)dimensionalities we obtained after performing SDNE + SVD method. The obtained datasets after performing PCA and SDNE + SVD were split into 70% training set, 10% validation set, and 20% testing set. 70% data (input features + label feature) were used to train the models to make them learn from the cases epoch-by-epoch, 10% data ((input features + label feature) were used to assess if the models are capable to perform well on those ‘unseen’ data, while the remaining 20% data (only input features) were used to feed into the well-trained models, their output results are then compared with the true label feature. Notably, we exclusively focus on predicting ‘generalized primary delay’ without engaging in any real-time prediction, primarily due to a lack of real-time information, including factors such as driver absence, incidents, weather, signal failures, and the like. Our approach focuses on forecasting a type of ‘generalized primary delay’ that is not specific to a particular day but is instead intended for broader use in the context of train design. This predictive capability proves valuable for anticipating potential delays in mid-term design scenarios. For instance, when a new timetable is devised, featuring trains slightly different from those in the past, our method can effectively forecast their generalized primary delays based on historical data, even if these new trains have not yet been implemented in practice.

Through the data pre-processing phase we discovered several quality issues that may have negative effects on prediction results (listed in Table 6). Overall, the statistical resource is an imbalanced dataset - similar to the electronic fraud dataset, in which fraudulent transactions are significantly lower than the normal healthy transactions. On the one hand, it is challenging to generate competitively stable and reliable prediction results with the limitation of highly imbalanced and small-scale dataset. On the other hand, it is more convincing if our proposed SDNE + SVD method shows its outperformed advantages on the imbalanced and small dataset over other competitive methods. Apart from this, several data quality issues such as noisy outliers, data incompleteness, and inconsistent format, were identified. According to the data requirements we expected, various policies for amending these issues have been performed (shown in the column of “Policy” in Table 6).

Fig. 6a illustrates the frequency distribution of primary delays, where severe delay and serious delay (more than 10 min) samples only account to around 9% of the total number of observations. From the perspective of passengers, whether/to what extent the train will delay is more mattering than knowing precisely how long the delay holds. One of the scopes of this study is to enhance the identification capability on rare minority classes rather than just achieving greater overall accuracy on test dataset. Considering these factors, we designed the train delay prediction task as a classification problem. In terms of various extent of delays, minor delays of a few minutes may be considered acceptable and might not significantly impact normal operations and passenger experience. Furthermore, there is established standards that specified in the Delay Attribution Report[4] by Rail Delivery Group to ORR (Office of Rail and Road), regarding what should be considered as an acceptable delay/moderate delay/and severe delay within the industrial sector in UK. We thus divided the delay cases into 5 categories: No delay (0), mild delay (0–3 min), moderate delay (3–10 min), serious delay (10–30 min), and severe delay (more than 30 min). Table 7 summarizes these five categories and provides index label for each category.

The original dataset is extremely imbalanced. The inherent drawbacks of Primary delay prediction problems make this dataset difficult to perform well on ML-based algorithms. The percentage of severe delay (more than 30 min) samples is significantly less than the rest of the categories (see Fig. 6a). In order to achieve a higher identification accuracy on minority classes, a re-sampling function SMOTE (Synthetic Minority Over-sampling Technique) was conducted before training (Chawla et al., 2002). SMOTE is able to generate new data examples by selecting the closest k data points in the feature space. The first step of working processes is to choose a random seed sample in minority classes and then to find k of the nearest neighbors for that data sample (shown in Fig. 6b). Finally, a synthetic example can be created at a random location between this data point and a randomly selected neighbor.

4.2. Node embedding implementation

In this sub-section we aim to evaluate the capability of our SDNE model by using the real-world railway network operated by TPE. At the first stage of implementation, the adjacency matrix A and weighted link matrix based on the network representation were generated and then used as the input of the SDNE model for iteratively training structural features. By optimizing the reconstruction loss, the local structural characteristics of the vertices can be preserved and $Y^{(k)}$ is the associated embedding representation. The model uses the 2nd-order loss function to make the embedding vectors close to the corresponding adjacent vertices, thereby preserving the global structural characteristics. The whole implementation was developed based on the open-source software library “Keras”,[5] which is a well-known deep learning API written in Python, running on the top of the machine learning platform “TensorFlow5”.[6] The dimension of vectors after embedding is 8 with 150 training epochs: alternative values of training epochs were tested (e.g., 120, 180) but the training loss curve shows the SDNE model was under-fitting/over-fitting at the end stage. Fig. 7a gives the information of the overall training loss (i.e., the L_{mix} defined in equation (8)) with the best parameter settings. And Fig. 7b shows the normalized last-layer

[4] https://www.orr.gov.uk/sites/default/files/2020-09/rdg-delay-attribution-review-report-2020-09-28.pdf.

[5] The latest official version of Keras can be accessed through: https://keras.io/.

[6] The official website of TensorFlow: https://www.tensorflow.org/.

Table 5
A list of the features used in modelling.

Feature Categories	Name of Feature	Dimensions			Explanation
		Original	PCA	SDNE + SVD	
Temporal Features	Date of Service	7	1	1	Mon., Tues., Wed., Thurs., Fri., Sat., Sun.
	Weekday/Holiday	2	1	1	Holidays or Working days
	Departure Time	1	4	4	The departure time from the origin station, calculated by the distance from 12pm midnight
	Arrival Time	1	4	4	The arrival time at the terminal station, calculated by the distance from 12pm midnight
Numerical Features	Passenger Volume	1	1	1	The total passenger flow volume of all the stations where the train stopped
	Total Margin	1	1	1	The total extra buffer time provided for making up the late running at all intermediate stations
	Speed Limit	4	2	2	The maximum speed the train can hold during running
	Link Travel Time	192	192	192×192	The average travel time on any observed edge in the network
Categorical Features	Rolling Stock Type	9	1	1	Which kind of locomotive the train employed
	Connectivity of Stations	177	177	177×177	Connectivity between any two stations in the network
Label Feature	Primary Delay	1191	5	5	Total primary delay time for the train service, categorized as 5 classes: None-delay, Mild delay, Moderate delay, Serious delay, and Severe delay
Network Features	Origin/Terminal/ Intermediate Station	177	50	3	The route matrix for each train service, including its origin/terminal and intermediate serving stations

Table 6
A summary of quality issues identified in the dataset.

Data quality issue	Corresponding features	Policy
Imbalanced and small dataset	Overall	SMOTE resampling strategy
Temporal features are not continuous	Departure Time Arrival Time	Cyclical variable projection –Polar coordinates systems
Noisy: containing outliers in Numerical features	Passenger Volume Total Margin	Z-score normalization (feature scaling)
Incomplete: lacking of certain values of interest in Network features	Terminal Station	Manually infer the missing value based on route information
Inconsistent: different formats in the same feature	Rolling Stock Type Route Stations	One-hot encoding

training loss curve over the 150 training epoches. All data were split into 3 parts: 70% as the training set, 20% testing set and 10% validation set.

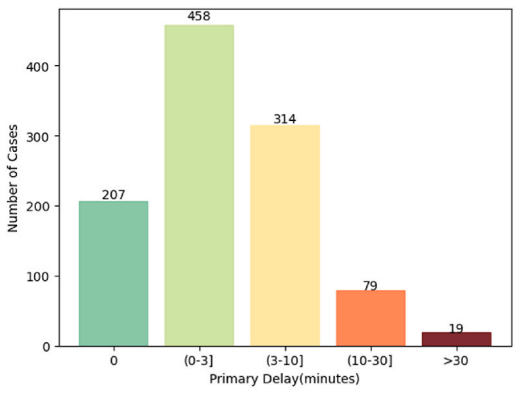
Then the node embedding for all stations were obtained, represented as a (177×8) matrix (Shown as Fig. 8), where $S = 177$ is the total number of stations in TPE network and $l = 8$ is the number of embedded dimensions for each station representation. Each row represents an embedding with the value between 0 and 1. Each column gives information about the value of a specific position (the 0^{th} to 7^{th} value). The reason of choosing 8 as the length of each embedding is because $2^8 = 256 >$ 177 (the number of nodes in this graph) $> 128 = 2^7$, In other words, '8' is the number of digits we decided to keep in each station embedding vector after using SDNE model, and each digit would be a float number values between 0 and 1. Imaging if we use one-hot vectors to represents all the 177 stations, only 8 digits are sufficiently capable (binary values at each digit). If 8 output dimensions are already enough for the simplest one-hot encoding, it should be enough for accommodating all the information in 177 station using SDNE. Which means it is sufficient to enclose all the necessary information using 8-digit embedding.

One general trend we can easily identify from the embedding heat-map (See Fig. 8) is that, the farther the geographical distance between two stations is (such as station 15 and station 34), the more distant the Euclidean distance between their embedded vectors would be, and vice versa. Moreover, the spatial iteration process among stations is illustrated through the gradient color distribution. In some parts of the heat-map, we can see that the embedding representations between adjacent stations are very similar but slightly different, which indicates that they are mapped to very close directions in the vector space. In the real-life network, the location of these stations are close to each other and thus their spatial distribution density is much higher than other areas (for example, Stations 0 to 10 as shown in Fig. 9 represent a set of closely embedded stations that all locating at the Great Manchester area).

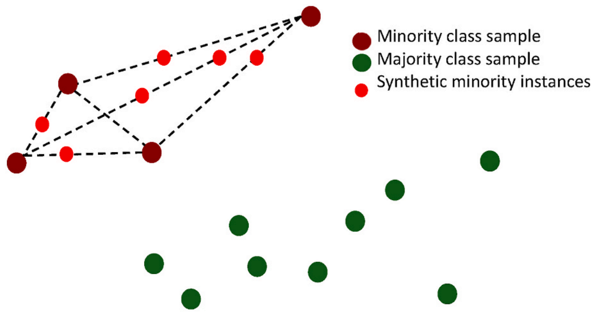
Therefore, from this perspective, the SDNE model is able to well preserve both the global and local characteristics of the stations in the TPE network.

4.3. Route embedding implementation

Having got the node embedding by SDNE, we continued to conduct route embedding. Using the SVD-based route embedding method (Algorithm 1), we encoded the route information for all the 1191 services found in the timetable. Note that there are many



(a) Distribution of Primary delays by minutes

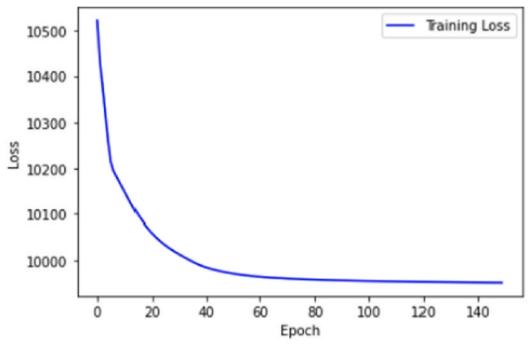


(b) Visualization of SMOTE resampling

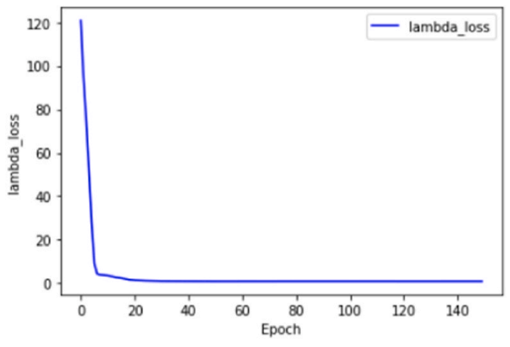
Fig. 6. Imbalanced dataset and the corresponding resampling strategy.

Table 7
Label processing plan.

Primary delay (mins)	Delay level description	Indexed level
0	None	0
(0, 3]	Mild delay	1
(3, 10]	Moderate delay	2
(10, 30]	Serious delay	3
>30	Severe delay	4



(a) The overall training losses over the whole epochs



(b) Normalised training losses

Fig. 7. Loss function curve of SDNE over training process.

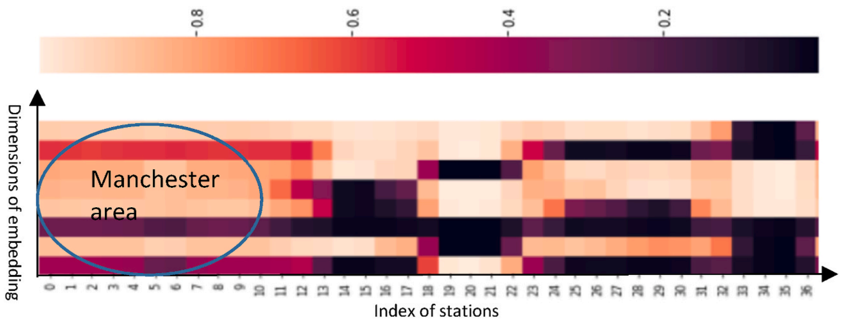


Fig. 8. Embedding vectors of 36 stations (out of 177) by SDNE.

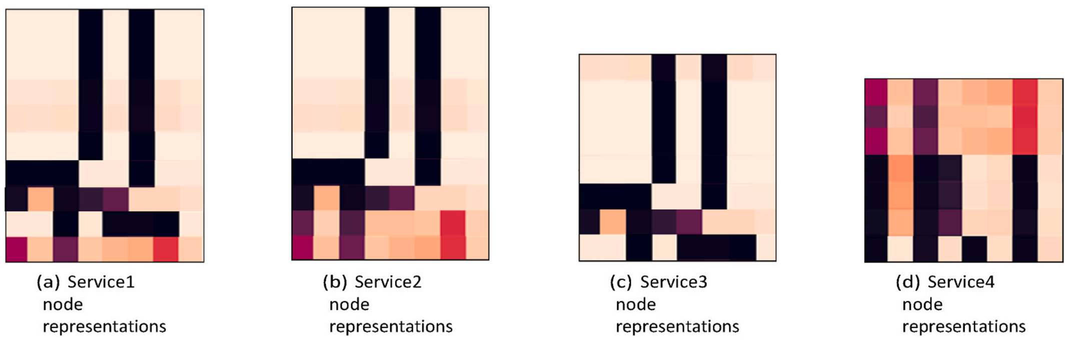


Fig. 9. Concatenated node embedding vectors for 4 different train services.

more train services than the possible routes as different trains can share the same route. As for determining r, the length of our route embedding vectors, we kept 90% of the entropy information in the original matrix $R(t)$. Based on the calculations, we find that the first three elements in Σ already preserve 93% of the essential energy of the whole node embedding vector. Therefore, we set $r = 3$ as the route embedding representation in our experiments.

Based on this approach, the final route embedding for each service was generated accordingly. As an illustrative example on the

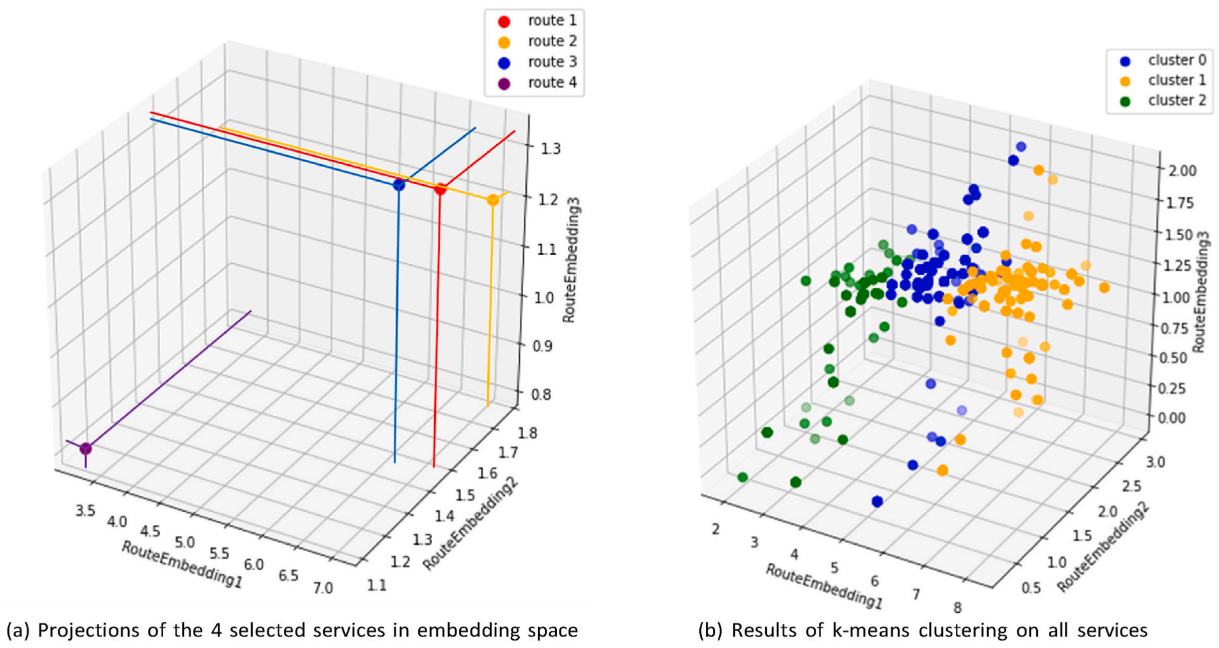


Fig. 10. 4 different service projections and the clustering results for all services.

effectiveness and accuracy of our SDNE + SVD method in generating route embedding vectors, 4 different services from the real network data have been selected with the aim of further validation:

Service1: Newcastle - Liverpool Lime Street: Newcastle, Chester-le-Street, Durham, Darlington, Northallerton, York, Leeds, Huddersfield, Manchester Victoria, Liverpool Lime Street.

Service2: Newcastle - Manchester Airport: Newcastle, Chester-le-Street, Durham, Darlington, Northallerton, York, Leeds, Huddersfield, Manchester Piccadilly, Manchester Airport.

Service3: Newcastle - Manchester Victoria: Newcastle, Durham, Darlington, Northallerton, York, Leeds, Huddersfield, Manchester Victoria.

Service4: Manchester Airport - Doncaster: Manchester Airport, Manchester Piccadilly, Stockport, Dore & Totley, Sheffield, Meadowhall, Doncaster.

Service 1 departures from Newcastle and terminal at Liverpool, as represented by one of the sub-route of the red line in Fig. 5. Service 2 is another service heading to Manchester Airport - very similar to the Service 1. These two services share most same dwell stations and are slightly different on the last two calling stations. Service 3 is a subset of Service 1: all the calling stations in Service 3 exist in Service 1 however it does not stop at some intermediate stations. Services 1–3 are all running on the route group labeled in red in Fig. 5. At last, Service 4 is totally different from Services 1–3, as it is a service selected from the route group represented by light blue route in Fig. 5. The sequentially concatenated station sequence representations for those four services shown as Fig. 9.

Fig. 10 depicts the location of these 4 services' route representations given by Algorithm 1 in $\mathbb{R}^3$. As we can see, the route embedding of Service 1 (red point) and Service 3 (blue point) are projected very closely. Followed by Service 2 (yellow point) - demonstrates the consistent pattern shown in the concatenated embedding. Compared to Service 3 (the subset of Service 1), it has a relatively larger Euclidean distance from Service 1 due to they have different serving stations at the end of journey. By contrast, Service 4 is embedded remotely from all other services above, because it is a service running within another route group. It can be observed that the SVD-based route embedding representations has highly preserved the characteristic of the network structures. Also, the essential information for each service has been uniformly condensed into a three dimensional vector, although they have different length and generated various sized matrices (i.e., 4 services shown in Fig. 9). Furthermore, as a strongly evidence of the accuracy of our route embedding approach (Algorithm 1), we clustered all the 1191 services represented by the 3-dimensional route vectors using the K-means method (Likas et al., 2003), and the result is shown in Fig. 10b. Each point in Fig. 10b represents different 'services' running in the three-corridor map (i.e., Fig. 5). These 'services' may only run single time during the observed period or multiple times. The clustering shows that all the 3d route embedding vectors have been automatically clustered into 3 different classes and each of them precisely corresponds to a route group shown in Fig. 5, where Clusters 0, 1 and 2 are associated with the route groups in purple, red and light blue respectively.

5. Delay prediction results

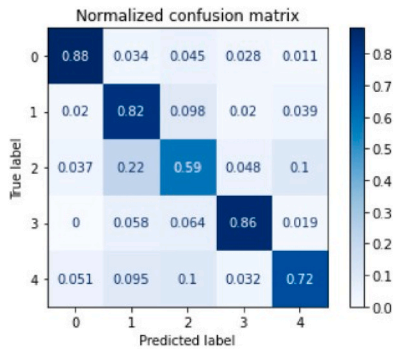
In this section, we report the delay prediction experiments. The 3d route-embedding was used as the feature for a train's route, plus other features given in Table 5 were fed into three classical supervised ML algorithms to predict primary train delays. The three ML algorithms are: Decision Tree (DT) (Quinlan, 1987); Random Forest (RF) (Ho, 1995); Multi-Layer Perceptron (MLP) (Gardner and Dorling, 1998). We examine the performance of our proposed SVD + SDNE framework with Principle Component Analysis (PCA) using these three standard ML prediction benchmarks. To assess the effectiveness of SDNE + SVD, we used the cross-validation provided in Table 8, confusion matrix (shown in Fig. 11), model accuracy (see Figs. 12 and 13), and measurements on prediction precision, recall, and F1 score (see Fig. 14).

In cross-validation, we randomly partitioned our data set into 5 parts (or folds), ran the learning experiments on each fold, and then computed the overall prediction power. Here, we adopt the standard measure on the prediction precision, defined as the ratio between total true-positive instances over all (true and false) positive ones (Tang et al., 2020). The results are presented in Table 8.

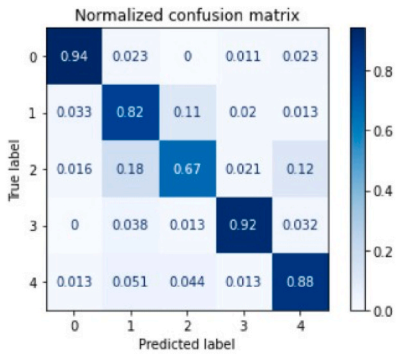
The table suggests that across all the five sets of data samples, the overall prediction precisions are highest with the MLP method, followed by RF, while DT achieved the lowest average precision scores. Comparing the two strategies, PCA and SDNE + SVD, the averages scores on all the three ML algorithms are similar. However, if we further check the standard deviation values on both strategies, the standard deviations from SDNE + SVD are consistently lower than that those with PCA. This result shows that the integration of SDNE and SVD is more suitable on unseen data instances than PCA in the tested cases on all the three ML algorithms. There is less likely that this model is overfitting during training. Furthermore, as the complexity of the prediction model increases, the prediction performances of our proposed SDNE + SVD become more stable and reliable.

Table 8
Comparison between PCA and SDNE + SVD method: 5-fold cross validation results.

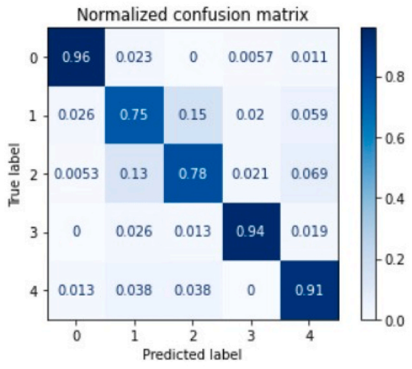
Strategy	Algorithm	1-fold	2-fold	3-fold	4-fold	5-fold	Average Score	Standard Deviation
PCA	DT	0.7198	0.7564	0.7347	0.7113	0.7381	0.7321	0.0227
	RF	0.7773	0.8190	0.8084	0.7916	0.8239	0.8040	0.0174
	MLP	0.8138	0.8378	0.8393	0.8150	0.8511	0.8314	**0.0146**
SDNE + SVD	DT	0.7443	0.7537	0.7035	0.7249	0.7132	0.7279	0.0187
	RF	0.8362	0.8338	0.8138	0.8196	0.7941	0.8195	0.0152
	MLP	0.8436	0.8181	0.8421	0.8346	0.8313	0.8339	**0.0091**



(a) Decision Tree Classifier



(b) Random Forest Classifier



(c) Multi-Layer Perceptron Classifier

Fig. 11. Confusion matrices on three different baselines.

Confusion matrix is another commonly used tool to visualize the performance of classification algorithms. Where the model accuracy is defined as the ratio of the number of correctly predicted instances to the total number of instances in the dataset. Accuracy can be written as below as equation (9), where TP are instances correctly predicted as positive, TN contains instances correctly predicted as negative, while FP are those instances incorrectly predicted as positive, and FN are instances incorrectly predicted as negative:

$$Accuracy = \frac{\sum_{k=1}^{K} TP_k + TN_k}{\sum_{k=1}^{K} (TP_k + TN_k + FP_k + FN_k)} \tag{9}$$

The confusion matrices from our experiments are shown in Fig. 11, where 11a, 11b and 11c are the confusion matrices generated by DT, RF and MLP, respectively. From the matrix of decision tree (Fig. 11a) we can see that this classifier performs well on no-delay (0) and serious delay (3) cases, where 88% and 86% accuracy outcomes were achieved respectively. However, about 41% of the moderate delay samples were classified incorrectly as other categories.

In the confusion matrix of random forest classifier (Fig. 11b), a significant improvement of accuracy can be seen on each delay category. Especially on no-delay (0), serious (3) and severe delay (4) samples, which reached 94%, 92% and 88% accuracy on test dataset, respectively. For the multi-layer perceptron classifier (Fig. 11c), the prediction accuracy on each sub-category is significantly boosted again, reflected in no-delay and serious delay cases with a higher level accuracy (96% and 94% respectively). It is worth noting that it reaches a better balance between different delay categories.

Fig. 12 illustrates the predictive performance of four distinct experimental benchmarks across various levels of delay. In each delay category (0–4), three consecutive bars are presented above, representing the predictive accuracy values of decision tree (DT) classifiers, random forest (RF) classifiers, and multilayer perceptron (MLP) under the corresponding delay category for each benchmark (from left to right). The three dashed lines in each subplot calculate the average accuracy of these classifiers across all delay classifications, representing the classifier's overall accuracy.

Observing these 4 bar graphs, substantial disparities in predictive accuracy are evident among different benchmarks in the prediction task due to various strategies applied for processing route features. Overall, the first benchmark (excluding route features from the model) results in the misclassification of almost all data points as either no delay or mild delay. This result makes it challenging to derive meaningful predictive insights, as the classification pattern appears to resemble with random chance.

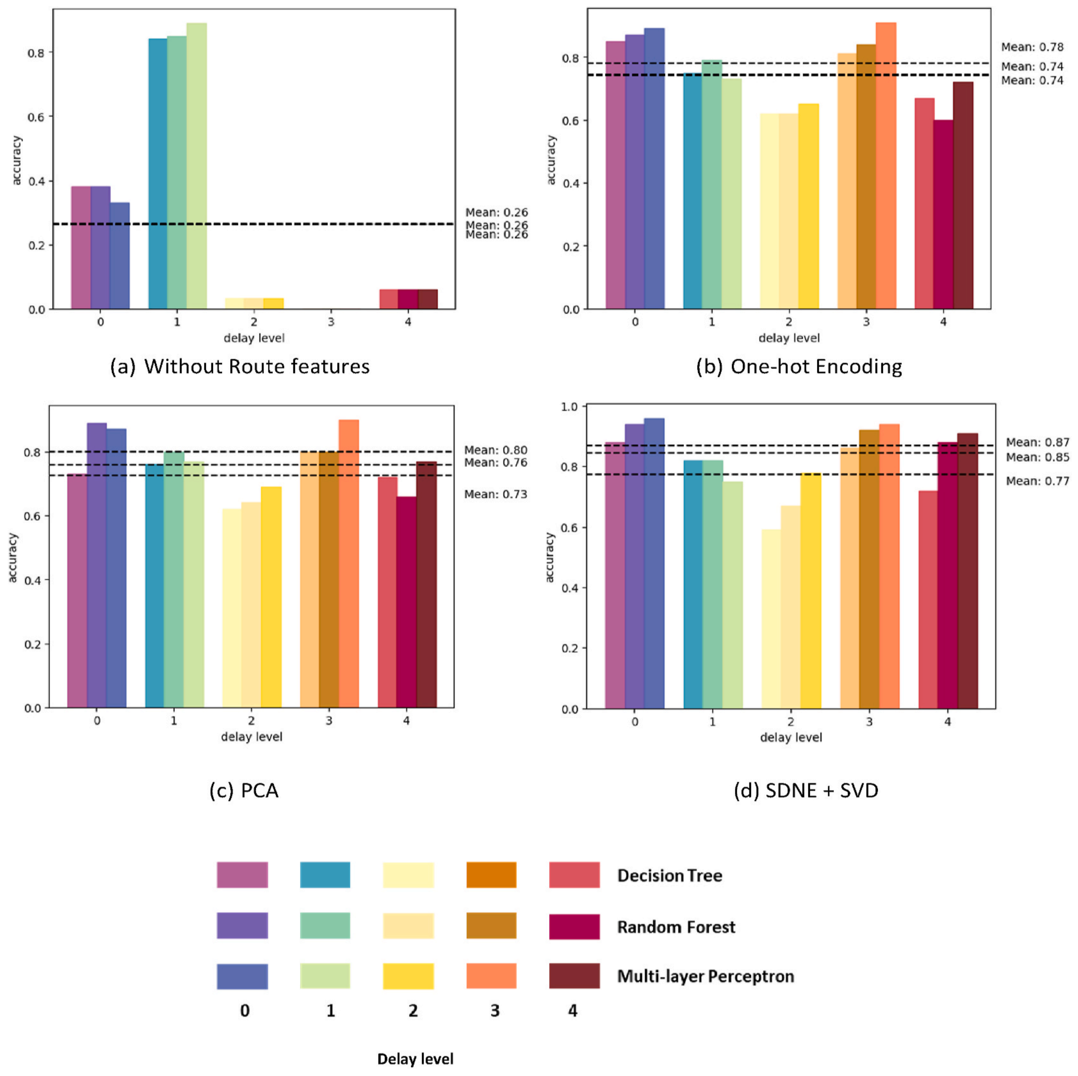


Fig. 12. Accuracy Comparison (on each delay level) between different benchmarks: Without Route features, One-hot Encoding, PCA, and SDNE + SVD.

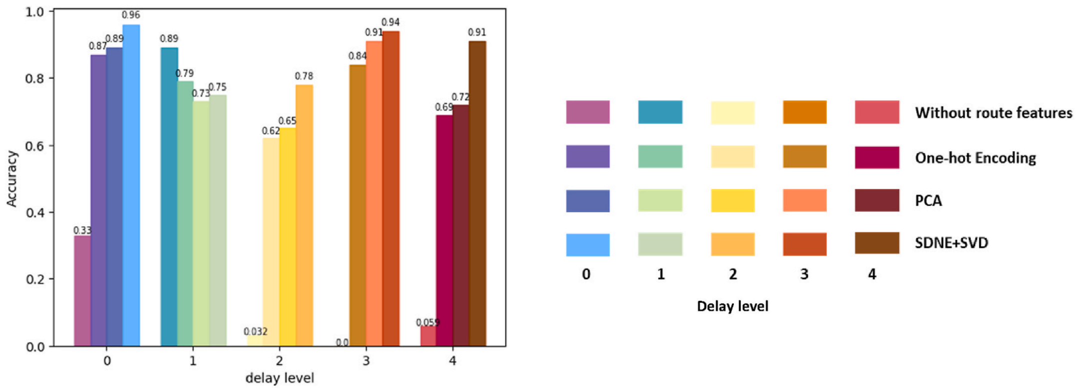


Fig. 13. Accuracy Comparison on the best performing predictor (MLP) among different benchmarks: Without Route features, One-hot Encoding, PCA, and SDNE + SVD.

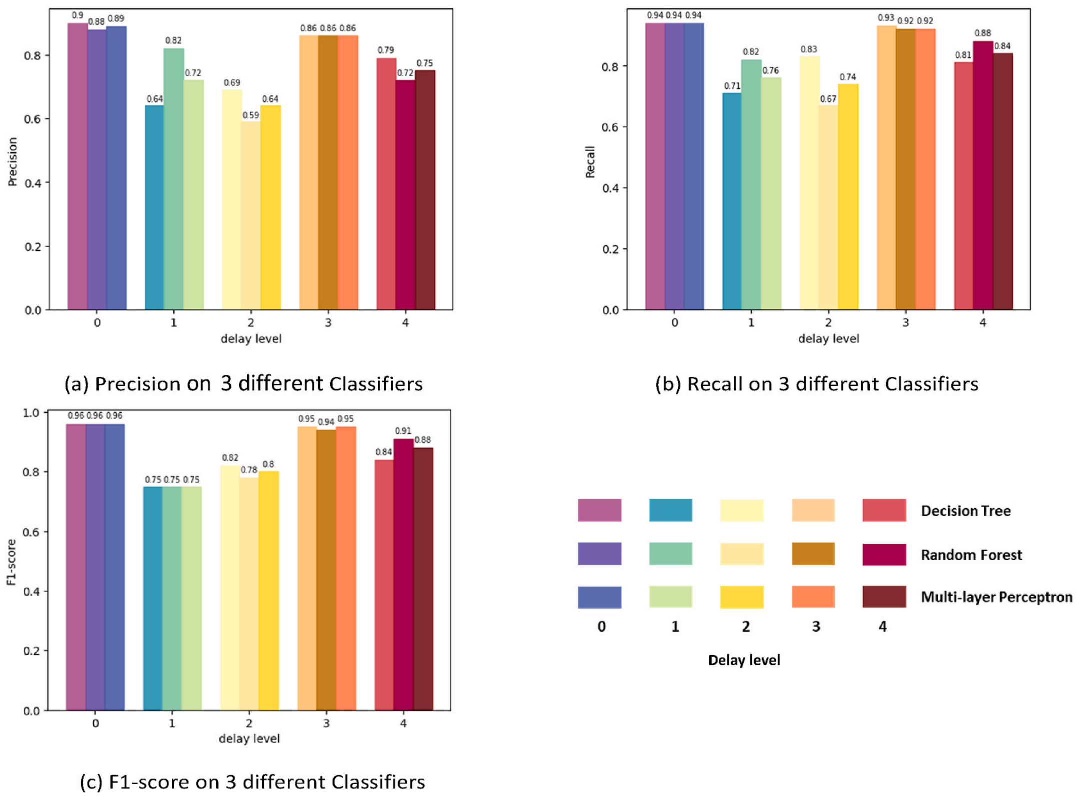


Fig. 14. Precision, Recall, and F1-score for 3 different classifiers on each delay level (SDNE + SVD).

Conversely, the remaining two benchmarks fairly demonstrate how the predictive accuracy gradually improves with the updates strategy for handling route features. From traditional methods (One-hot Encoding) to Auto-encoders (PCA), as reflected across all three classifiers. Notably, in our proposed framework, SDNE + SVD, when horizontally compared to its two competing benchmarks, all three classifiers exhibit superior predictive performance and higher average accuracy over five delay categories.

Furthermore, the histograms suggest that different classifiers hold various prediction capability on particular delay level. For example, if we pick up our SDNE + SVD model, apart from the mild delay, MLP classifier is consistently outperforming than other two classifiers. Another trend we can find is: compare to DT, MLP classifier receives a more stable and reliable prediction accuracy with lower performance fluctuation among all five classes, but the average accuracy is higher (87%). In order to furtherly make comparison explicitly over these four benchmarks, we exclusively choose the best performing predictor (MLP) for discussion in Fig. 13.

The histogram in Fig. 13 gives the information about to what extent the proposed SDNE + SVD framework achieve a better performance over other benchmarks (MLP as example). On each delay level, from left to right, bars with different colors denotes the prediction accuracy on the applied strategies of ‘Without Route features’, ‘One-hot Encoding’, ‘PCA’, and ‘SDNE + SVD’, respectively. From the bar chart we can see, SDNE + SVD is consistently outperforming over other three benchmarks on delay levels of (0), (2), (3) and (4), while maintain a similar accuracy result on mild delay (1) with other two competitive models. Especially, SDNE + SVD demonstrate it advantages on predicting the cases of moderate delay and severe delay. Due to these two categories are more likely to be wrongly classified as neighbor categories by the benchmarks.

Since all the candidate classifiers are supervised learning-based model, we use three different measures of effectiveness, namely, Precision, Recall, and F1-score, for further validating the performance of SDNE + SVD model on each delay category. Before that, we need to briefly give the definitions of these three measures as equation (10)–(12):

$$P_{(precision)} = \frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K} (TP_k + FP_k)} \tag{10}$$

$$P_{(recall)} = \frac{\sum_{k=1}^{K} TP_k}{\sum_{k=1}^{K} (TP_k + FN_k)} \tag{11}$$

Recall is a ratio represents the fraction of correctly predicted instances over the number of the ground truth positives cases, while the precision indicates the proportion that correctly predicted instances take account the overall predicted positive observations. The relationship between them is one increases then another one drops. The assessed model would under-perform if either of them is significantly lower than another one. In practice we generally want to achieve a trade-off between them. F1 score is then introduced for further evaluation and validation. F1 score conveys the balance between the precision and the recall and it can be computed by the following formula.

$$F1_{(score)} = \frac{2 \times P \times R}{P + R} \tag{12}$$

Fig. 14 gives the information about to what extent each category (i.e., None delay (0), Mild delay (1), Moderate delay (2), Serious delay (3) and Severe delay (4)) has been effectively predicted on the three different baseline models. Precision calculates how many train services are correctly predicted among all predictions made on this category. Recall tells us how many instances have been correctly identified among all real train services that should belong to this category. In fact, we are aim to reduce relative impacts of both these two types of errors on our use-case. Such that a model with higher F1-score is more desirable in our perspective. The less of differences between Precision and Recall among the predicted categories, the better the algorithm is. From this perspective, the Multi-Layer Perceptron Classifier perform perfectly on balancing Precision and Recall. On the opposite, Decision Tree and Random Forest Classifiers are more fluctuated on the Precision/Recall measures.

Table 9 shows that our method achieved an improvement in overall prediction accuracy over the baselines in the dataset, and reached 86.80% on the best performing model. Fig. 15 gives the information about the overall training time of our method is significantly lower than that of competitive method on every benchmark, and a trend of huge decline of necessary computational efforts after implementing SDNE + SVD method can be seen. The model running time efficiency on the most accurate predictor (MLP) is improved on the level of 490% - from 1417s to 289s - see the green line in Fig. 15. Which means the delay status of each unknown train service is promisingly to be calculated within 0.24s, even if we migrate it to a brand new dataset and the model needs to be re-trained.

6. Conclusions and future work

In this paper, we proposed a novel method, namely "SDNE + SVD Embedding", to accurately represent train routes as short vectors (of length 3) as an important train feature to further conduct primary delay prediction using supervised ML. It is the first time that train routes (as paths in a graph) are embedded for the purpose of delay prediction. To capture the highly complex network topology information of the considered railway network, we use first-order and second-order proximity to characterize the local and global network structure. The embedded station representations are able to preserve both local and global structures. We empirically evaluated the generated route representations in a real-world example based on the rail network of TransPennine Express. The results show that our model outperforms the competitive method PCA, plain One-hot Encoding in terms of overall prediction accuracy and total training time.

Based on the experiments conducted on services extracted from a real-world train network, the following observations were delineated: Firstly, it is believed that the provided operational records and primary delay data exhibit superior fitting capabilities when they are fed to more intricate ML models such as MLP and RF. Subsequently, the implementation of SDNE in conjunction with SVD for route embedding yields comparatively enhanced prediction accuracy, specifically in predicting the no-delay instances and serious/ severe delay cases, as opposed to the utilization of PCA. The overall predictive accuracy exhibited an incremental trend in the implementation of SDNE + SVD. Thirdly, the temporal requirement for the convergence of the ML models has been substantially diminished thanks to SDNE + SVD. It generally took over 20 min for training the models if we used the PCA method. While the training time decreases to less than 5 min using SDNE + SVD.

An important aspect that cannot be ignored is that more insights of the vital element of connectivity between routes that share common transfer stations should be introduced in this study. Unfortunately, due to the current study is completely constructed upon the acquired TPE dataset and they failed to provide us more information about transfer stations and passengers who transfer at these vital connectivity stations. Additionally, sometimes, passengers may also would like to know how much delay their train will experience at a particular station. However, the current work is limited to predicting the generalized delay for a train. Various real-world network features, such as historical delay times at every station a train passed, should be included to predict delay time at intermediate stations. Moreover, the current work only focuses on primary delays, however, exploring delay propagation mechanisms between different train services is important in explaining the determinants of delay occurrence (Huang et al., 2024). In future work we would aim to 1) consider the significance of those transfer stations and delve deeper into transfer station dynamics, by taking the best advantage of the available dataset where possible, 2) further refine the granularity of forecast results to transition from crude estimation to more precise and customized prediction, 3) As the experiment results shows, current study that utilized graph-embedding

Table 9
Comparison between PCA and SDNE + SVD method: Overall accuracy.

Strategy	PCA			SDNE + SVD		
Algorithm	DT	RF	MLP	DT	RF	MLP
Overall Accuracy	76.68%	82.09%	83.89%	77.40%	84.59%	**86.80%**

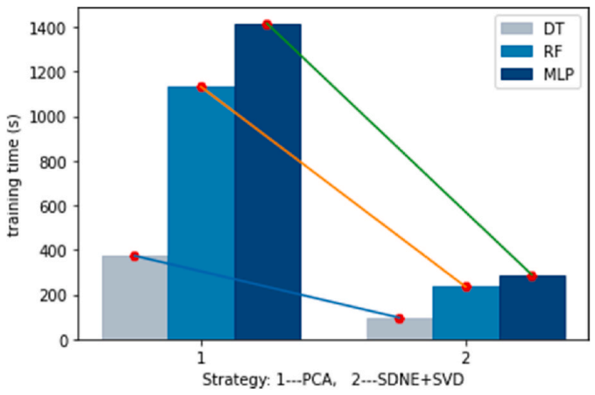


Fig. 15. Comparison between PCA and SDNE + SVD method: training time reduction.

techniques and ML predictors demonstrated the feasibility and effectiveness of incorporating both 'route-based' characteristics and historical train service delay records (i.e., origin, destination, departure/arrival times, rolling stock type etc.) into the 'generalized primary delay' prediction framework. However, due to the nature of all the primary delays are highly incident-dependent – We appreciate the potential benefits of incorporating additional incident factors and weather to further enhance our predictions. To refine this, we have decided to include some of these factors (i.e., weather, failure in equipment or infrastructure) into our future secondary delay paradigms rather than in the current study. Specifically, we intend to look into the correlations between primary delays and secondary delays, and the propagation patterns of secondary delays using graph embedding based approaches.

Declaration of competing interest

I, Ruifan Tang, declare that I have no financial or personal relationships with any individuals or organizations that could inappropriately influence or bias the content of this submitting paper. Furthermore, I have no conflicts of interest to declare with regards to the research presented in this paper.

I confirm that the research presented in this paper was conducted in an impartial and objective manner, and that the findings and conclusions drawn from this research are based solely on the data and analysis presented in the paper. All sources of data for this research have been disclosed and are acknowledged appropriately in the paper.

I understand that any undisclosed conflicts of interest could undermine the integrity and credibility of the research presented in this paper. As such, I have taken great care to ensure that this declaration of interest statement accurately reflects my relationship with any individuals or organizations that may have a stake in the research presented in this paper.

Acknowledgement

The authors hereby would like to thank First Group TransPennine Express for providing the train delay and timetable data used in this study.

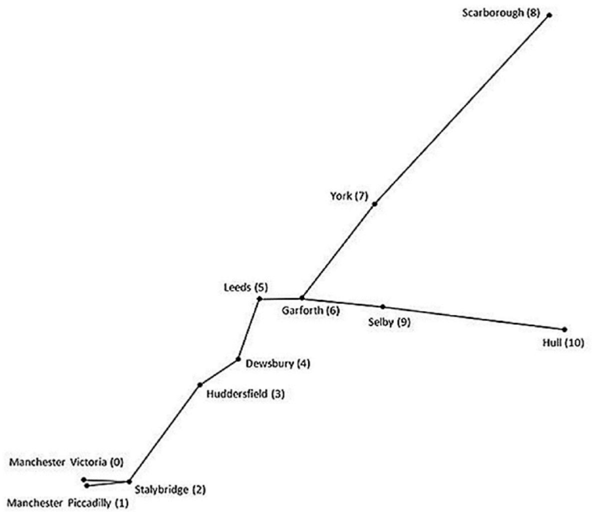
Appendix

Conventional Feature Engineering Techniques

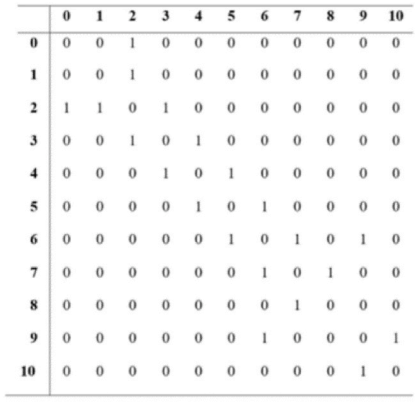
In Section 2, we introduced that graph embedding is a powerful and versatile technology for dimensionality reduction. In fact, apart from the graph embedding techniques, two traditional and more generic dimensionality reduction techniques have been commonly used in feature engineering processes - One-hot encoding (Harris and Harris, 2010) and Principle Component Analysis (PCA) (Wold et al., 1987). Most machine learning models require the input data to be processed to a matrix format (known as "tensor") before it can be fit to them. One-hot encoding is a method of converting categorical data into matrix format such that the data can be directly understood by machines. Each categorical value is transformed into a new categorical column under one-hot encoding, and each column is given a binary value of 1 or 0. The index is designated with a 1 and all of the remaining values are 0. An example is provided in Fig. 16: As we can see, 11 nodes and 10 edges are in this network. Each station (node) is assigned a unique number beside them. Black nodes in this graph represents stations and the lines connect two stations represent the tracks between them (see Fig. 16a. Single-line or multi-line railway is considered as one edge in this model). The spatial relationships between these stations are shown as well. In the one-hot matrix on the right hand side (see Fig. 16b), a '1' represents that there is a visible edge between the corresponding two stations while 0 indicates no connection. And then the topology of this network can be described in the matrix form.

However, in most circumstances, one-hot encoding is not an optimal solution for representing network features due to the curse of dimensionality (Indyk and Motwani, 1998). Imagine a given network with hundreds/thousands of railway stations and edges.

Apparently it is not desirable to represent all the nodes and connectivity status with a one-hot matrix of excessive dimensions but sparse on the essential information. Firstly, if we have more features than observations, we run the risk of massively overfitting our model and this may result in poor sample performance at the end. Secondly, when there are too many features, outputs of the ML-based models may become harder to cluster. That is, too many dimensions cause every encoded station to appear equidistant from each other. Then all the observations appear equally alike and no meaningful clusters can be formed. Lastly, with the growth of the size of encoded matrix, computational efficiency and feasibility will decrease. Due to its poor performance in large-scale networks and the potential risk of dimensionality explosion, we will not implement this encoding strategy on network topology features. Instead, we find that applying one-hot encoding only to those low-dimensional categorical features such as “Type of Rolling Stock” will be still acceptable.



(a) an 11-node network



	0	1	2	3	4	5	6	7	8	9	10
0	0	0	1	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0
2	1	1	0	1	0	0	0	0	0	0	0
3	0	0	1	0	1	0	0	0	0	0	0
4	0	0	0	1	0	1	0	0	0	0	0
5	0	0	0	0	1	0	1	0	0	0	0
6	0	0	0	0	0	1	0	1	0	1	0
7	0	0	0	0	0	0	1	0	1	0	0
8	0	0	0	0	0	0	0	1	0	0	0
9	0	0	0	0	0	0	1	0	0	0	1
10	0	0	0	0	0	0	0	0	0	1	0

(b) one-hot representation for the 11-node network

Fig. 16. The process of converting network nodes into one-hot matrix.

PCA is defined as a technique for data analysis and pre-processing. It converts the original data into a set of linearly independent representations on each dimension by linear transformation algorithm. And it is commonly used to extract the main feature components of data vectors and thus reduce data dimensions. Table 10 below gives the PCA algorithm we applied in this study:

From the actions of calculating eigenvalues into a separate matrix and taking the highest d dimensions we can see, PCA is characterized as an orthogonal linear transformation that shifts the data into a new coordinate system such that the largest variance by some scalar projection of the data comes to lie on the first coordinate (referred to as the first principal component), the second largest variance on the second coordinate, and so on. From this perspective, as a rather good feature engineering technique, PCA already been successfully applied to dimension reduction and performance improvement on the task of image classification (e.g., Nasution et al., 2018; Uddin et al., 2021). Although PCA greatly avoids the curse of dimensionality and improves the computation speed compared to one-hot encoding, one of the drawbacks is that the influence power of outliers will be magnified and the model is likely to overfit. Similar with the pruning challenge in decision tree (Wang et al., 2010), how we reduce the dimensions into a reasonable range (neither too wide or small) is a challenging task - if it is too large then the potential of this algorithm would be wasted but if it too small then PCA would not capture enough information and its power will be undermined. Another disadvantage of PCA in our route representation problem is its poor interpretability, since data processed by PCA often lack intuitive meanings. It is found that our graph embedding based approach offers considerably better interpretability as evidenced by K-means clustering (see Section 4.3), where each route under our embedding is precisely clustered as a point reflecting the similarities of the routes.

Table 10
Algorithm 2: Principle Component Analysis

```
Input: Training set $D = \{x_1, x_2, \ldots, x_m\}$, where each sample has $n$ dimensions. Target Dimension $d$
    Output: Transformed Matrix $W$
    Transpose $D$ to $X$, where the shape of $X$ is $n \times m$
    for $i = 0$ to $n$
        for each $x_{ij}$ in $X$
            $x_{ij} = x_{i1} + x_{i2} + \ldots + x_{im}/m$
        end for
    end for
    Calculate the covariance matrix $C = XX^T/m$
    Calculate the eigenvalues of the covariance matrix $C$ and the corresponding feature vector
```

(continued on next page)

Table 10 (*continued*)

Arrange the eigenvectors into a matrix in rows according to the size of the corresponding eigenvalues
Take the first d rows as matrix P
$W = PX$

Introduction of Singular Value Decomposition

SVD performs the matrix decomposition by extracting the most essential information (called 'singular values') in the original vector. By using SVD we are able to represent the original dataset by a much smaller dataset, such that the noise and redundant information are significantly reduced. From this perspective, SVD can be regarded as a process of extracting the most relevant features from a set of ordered node embedding. SVD decomposes the original matrix D into three separate sub-matrices: U, Σ and V^T, with the size of $m \times m$, $m \times n$ and $n \times n$, respectively:

$$D_{m\times n} = U_{m\times m} \bullet \Sigma_{m\times n} \bullet V^T_{n\times n} \tag{13}$$

The decomposition process above will generate a matrix Σ with only diagonal elements (see Fig. 17). One of important characteristics of this matrix is that its diagonal elements are arranged from the largest value to the smallest. Similar to the eigenvalues and eigenvectors we introduced in PCA method, these diagonal elements are the singular values we aim to obtain, and they correspond to the singular values (highly condensed essential information) in the original data matrix. We can also find that after the r-th singular value (where $r < n$ is the number of non-zero values in Σ), other following singular values are all set to 0. This means that there are only r important features in the original matrix D, while the rest of the features are all noise or redundant features.

Fig. 17 illustrates the process of our route embedding approach using SVD. The singular values can be interpreted as a measure of the total essential information contained in the matrix $R(t)$. In this case, each $R(t)$ is equivalent to concatenated sequential node embedding representations (i.e., the right hand side matrix of Fig. 3). SVD breaks down this matrix into smaller components. These singular values represent the most important parts of the original matrix, and they contain all the essential information needed to reconstruct the whole matrix.

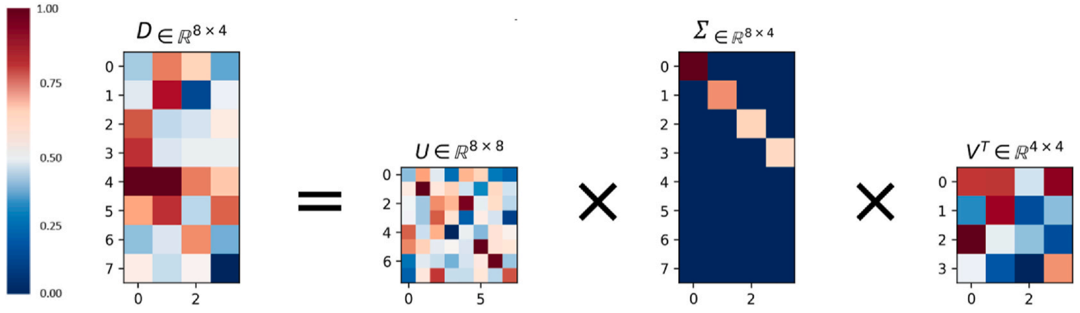


Fig. 17. Visualization of Singular Value Decomposition where m = 8 and n = 4.

Data availability

The authors do not have permission to share data.

References

Abdi, H., Williams, L.J., 2010. Principal component analysis. Wiley interdisciplinary reviews: Comput. Stat. 2, 433–459.
Ahmed, A., Shervashidze, N., Narayanamurthy, S., Josifovski, V., Smola, A.J., 2013. Distributed large-scale natural graph factorization. In: Proceedings of the 22nd International Conference on World Wide Web, pp. 37–48.
Artan, M.Ş., Şahin, İ., 2023. A stochastic model for reliability analysis of periodic train timetables. Transportmetrica B: Transport Dynamics 11, 572–589.
Banerjee, M., Pal, N.R., 2014. Feature selection with svd entropy: some modification and extension. Inf. Sci. 264, 118–134.
Barbour, W., Mori, J.C.M., Kuppa, S., Work, D.B., 2018. Prediction of arrival times of freight traffic on us railroads using support vectorregression. Transport. Res. C Emerg. Technol. 93, 211–227.
Belkin, M., Niyogi, P., 2001. Laplacian eigenmaps and spectral techniques for embedding and clustering. Adv. Neural Inf. Process. Syst. 14.
Bengio, Y., et al., 2009. Learning deep architectures for ai. Foundations and trends® in Machine Learning 2, 1–127.
Büker, T., Seybold, B., 2012. Stochastic modelling of delay propagation in large networks. Journal of Rail Transport Planning & Management 2, 34–50.
Cao, S., Lu, W., Xu, Q., 2015. Grarep: learning graph representations with global structural information. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 891–900.
Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P., 2002. Smote: synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357.
Chen, Z., Wang, Y., 2019. Impacts of severe weather events on high-speed rail and aviation delays. Transport. Res. Transport Environ. 69, 168–183.
Corman, F., D'Ariano, A., Hansen, I.A., 2014. Evaluating disturbance robustness of railway schedules. Journal of Intelligent Transportation Systems 18, 106–120.
Corman, F., Kecman, P., 2018. Stochastic prediction of train delays in real-time using bayesian networks. Transport. Res. C Emerg. Technol. 95, 599–615.

Cui, P., Wang, X., Pei, J., Zhu, W., 2018. A survey on network embedding. IEEE Trans. Knowl. Data Eng. 31, 833–852.

Gardner, M.W., Dorling, S., 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric environment 32, 2627–2636.

Gaurav, R., Srivastava, B., 2018. Estimating train delays in a large rail network using a zero shot markov model. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC). IEEE, pp. 1221–1226.

Ghaemi, N., Cats, O., Goverde, R.M., 2017. Railway disruption management challenges and possible solution directions. Public Transport 9, 343–364.

Gorman, M.F., 2009. Statistical estimation of railroad congestion delay. Transport. Res. E Logist. Transport. Rev. 45, 446–456.

Goverde, R.M., 2010. A delay propagation algorithm for large-scale railway traffic networks. Transport. Res. C Emerg. Technol. 18, 269–287.

Goverde, R.M., Corman, F., D'Ariano, A., 2013. Railway line capacity consumption of different railway signalling systems under scheduled and disturbed conditions. Journal of rail transport planning & management 3, 78–94.

Grover, A., Leskovec, J., 2016. node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864.

Hamilton, W.L., Ying, R., Leskovec, J., 2017. Representation learning on graphs: methods and applications. arXiv preprint arXiv:1709.05584.

Harris, D., Harris, S.L., 2010. Digital Design and Computer Architecture. Morgan Kaufmann.

Heglund, J.S., Taleongpong, P., Hu, S., Tran, H.T., 2020. Railway delay prediction with spatial-temporal graph convolutional networks. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). IEEE, pp. 1–6.

Higgins, A., Ferreira, L., Lake, M., 1999. Scheduling rail track maintenance to minimise overall delays. In: Transportation and Traffic Theory: Proceedings of the 14th International Symposium on Transportation and Traffic Theory. Elsevier Science Ltd., pp. 779–796

Hinton, G., Deng, L., Yu, D., Dahl, G.E., Mohamed, A.r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Sainath, T.N., et al., 2012. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. IEEE Signal Process. Mag. 29, 82–97.

Ho, T.K., 1995. Random decision forests. In: Proceedings of 3rd International Conference on Document Analysis and Recognition. IEEE, pp. 278–282.

Huang, P., Guo, J., Liu, S., Corman, F., 2024. Explainable train delay propagation: a graph attention network approach. Transport. Res. E Logist. Transport. Rev. 184, 103457.

Huang, P., Lessan, J., Wen, C., Peng, Q., Fu, L., Li, L., Xu, X., 2020. A bayesian network model to predict the effects of interruptions on train operations. Transport. Res. C Emerg. Technol. 114, 338–358.

Huang, P., Li, Z., Wen, C., Lessan, J., Corman, F., Fu, L., 2021. Modeling train timetables as images: a cost-sensitive deep learning framework for delay propagation pattern recognition. Expert Syst. Appl. 177, 114996.

Huang, P., Wen, C., Peng, Q., Jiang, C., Yang, Y., Fu, Z., 2019. Modeling the influence of disturbances in high-speed railway systems. J. Adv. Transp. 2019.

Huang, P., Spanninger, T., Corman, F., 2022. Enhancing the understanding of train delays with delay evolution pattern discovery: a clustering and bayesian network approach. IEEE Trans. Intell. Transport. Syst. 23, 15367–15381.

Indyk, P., Motwani, R., 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pp. 604–613.

Kang, Z., Xu, H., Hu, J., Pei, X., 2019. Learning dynamic graph embedding for traffic flow forecasting: A graph self-attentive method. In: 2019 IEEE intelligent transportation systems conference (ITSC). IEEE, pp. 2570–2576.

Kecman, P., Corman, F., Meng, L., 2015. Train delay evolution as a stochastic process. In: 6th International Conference on Railway Operations Modelling and Analysis (RailTokyo2015), IVT. ETH Zurich; Orange Labs.

Kecman, P., Goverde, R.M., 2014. Online data-driven adaptive prediction of train event times. IEEE Trans. Intell. Transport. Syst. 16, 465–474.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 25.

Lessan, J., Fu, L., Wen, C., 2019. A hybrid bayesian network model for predicting delays in train operations. Comput. Ind. Eng. 127, 1214–1222.

Li, D., Daamen, W., Goverde, R.M., 2016. Estimation of train dwell time at short stops based on track occupation event data: a study at a Dutch railway station. J. Adv. Transp. 50, 877–896.

Li, Z., Wen, C., Hu, R., Xu, C., Huang, P., Jiang, X., 2021. Near-term train delay prediction in the Dutch railways network. International Journal of Rail Transportation 9, 520–539.

Likas, A., Vlassis, N., Verbeek, J.J., 2003. The global k-means clustering algorithm. Pattern Recogn. 36, 451–461.

Ludvigsen, J., Klæboe, R., 2014. Extreme weather impacts on freight railways in europe. Nat. Hazards 70, 767–787.

Magadagela, K., Nel, H., Marnewick, A., 2017. Identification of delay factors that affect high dwell times of freight trains. In: 2017 IEEE Technology & Engineering Management Conference (TEMSCON). IEEE, pp. 179–184.

Marković, N., Milinković, S., Tikhonov, K.S., Schonfeld, P., 2015. Analyzing passenger train arrival delays with support vector regression. Transport. Res. C Emerg. Technol. 56, 251–262.

Meester, L.E., Muns, S., 2007. Stochastic delay propagation in railway networks and phase-type distributions. Transp. Res. Part B Methodol. 41, 218–230.

Mou, W., Cheng, Z., Wen, C., 2019. Predictive model of train delays in a railway system. In: RailNorrköping 2019. 8th International Conference on Railway Operations Modelling and Analysis (ICROMA), Norrköping, Sweden, June 17th–20th, 2019. Linköping University Electronic Press, pp. 913–929.

Nair, R., Hoang, T.L., Laumanns, M., Chen, B., Cogill, R., Szabó, J., Walter, T., 2019. An ensemble prediction model for train delays. Transport. Res. C Emerg. Technol. 104, 196–209.

Nasution, M., Sitompul, O., Ramli, M., 2018. Pca based feature reduction to improve the accuracy of decision tree c4. 5 classification. In: Journal of Physics: Conference Series. IOP Publishing, 012058.

Nilsson, R., Henning, K., 2018. Predictions of Train Delays Using Machine Learning.

Olsson, N.O., Haugland, H., 2004. Influencing factors on train punctuality—results from some Norwegian studies. Transp. Policy 11, 387–397.

Oneto, L., Fumeo, E., Clerico, G., Canepa, R., Papa, F., Dambra, C., Mazzino, N., Anguita, D., 2018. Train delay prediction systems: a big data analytics perspective. Big data research 11, 54–64.

Perozzi, B., Al-Rfou, R., Skiena, S., 2014. Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710.

Peters, J., Emig, B., Jung, M., Schmidt, S., 2005. Prediction of delays in public transportation using neural networks. In: International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06). IEEE, pp. 92–97.

Plummer, M.D., 2007. Graph factors and factorization: 1985–2003: a survey. Discret. Math. 307, 791–821.

Quinlan, J.R., 1987. Simplifying decision trees. Int. J. Man Mach. Stud. 27, 221–234.

Rissanen, J., 1978. Modeling by shortest data description. Automatica 14, 465–471.

Robert, N., Kim, H., 2018. Predictions of Train Delays Using Machine Learning. Examensarbete Inom Datateknik, KTH Stockholm.

Rößler, D., Reisch, J., Hauck, F., Kliewer, N., 2021. Discerning primary and secondary delays in railway networks using explainable ai. Transp. Res. Procedia 52, 171–178.

Şahin, İ., 2017. Markov chain model for delay distribution in train schedules: assessing the effectiveness of time allowances. Journal of rail transport planning & management 7, 101–113.

Shen, Y., Jin, C., Hua, J., Huang, D., 2020. Ttpnet: a neural network for travel time prediction based on tensor decomposition and graph embedding. IEEE Trans. Knowl. Data Eng. 34, 4514–4526.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A.Y., Potts, C., 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In: Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pp. 1631–1642.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q., 2015. Line: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1067–1077.

Tang, T., Liu, R., Choudhury, C., 2020. Incorporating weather conditions and travel history in estimating the alighting bus stops from smart card data. Sustain. Cities Soc. 53, 101927.

Tanwar, S., Ramani, T., Tyagi, S., 2018. Dimensionality reduction using pca and svd in big data: a comparative case study. In: Future Internet Technologies and Trends: First International Conference, ICFITT 2017, Surat, India, August 31-September 2, 2017, Proceedings 1. Springer, pp. 116–125.

Tiong, K.Y., Ma, Z., Palmqvist, C.W., 2022. Prediction of real-time train arrival times along the Swedish southern mainline. In: 18th International Conference on Railway Engineering Design and Operation, COMPRAIL 2022. WIT Press, pp. 135–143.

Uddin, M.P., Mamun, M.A., Hossain, M.A., 2021. Pca-based feature reduction for hyperspectral remote sensing image classification. IETE Tech. Rev. 38, 377–396.

Ulak, M.B., Yazici, A., Zhang, Y., 2020. Analyzing network-wide patterns of rail transit delays using bayesian network learning. Transport. Res. C Emerg. Technol. 119, 102749.

Wang, D., Cui, P., Zhu, W., 2016. Structural deep network embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1225–1234.

Wang, P., Zhang, Q.p., 2019. Train delay analysis and prediction based on big data fusion. Transportation Safety and Environment 1, 79–88.

Wang, T., Qin, Z., Jin, Z., Zhang, S., 2010. Handling over-fitting in test cost-sensitive decision tree learning by feature selection, smoothing and pruning. J. Syst. Software 83, 1137–1147.

Wen, C., Mou, W., Huang, P., Li, Z., 2020. A predictive model of train delays on a railway line. J. Forecast. 39, 470–488.

Weng, J., Zheng, Y., Yan, X., Meng, Q., 2014. Development of a subway operation incident delay model using accelerated failure time approaches. Accid. Anal. Prev. 73, 12–19.

Wold, S., Esbensen, K., Geladi, P., 1987. Principal component analysis. Chemometr. Intell. Lab. Syst. 2, 37–52.

Yaghini, M., Khoshraftar, M.M., Seyedabadi, M., 2013. Railway passenger train delay prediction via neural network model. J. Adv. Transp. 47, 355–368.

Yang, D., Ma, Z., Buja, A., 2011. A sparse svd method for high-dimensional data. arXiv preprint arXiv:1112.2433.

Yang, Y., Huang, P., Peng, Q., Li, J., Wen, C., 2019. Statistical delay distribution analysis on high-speed railway trains. Journal of Modern Transportation 27, 188–197.

Yuan, J., Hansen, I.A., 2007. Optimizing capacity utilization of stations by estimating knock-on train delays. Transp. Res. Part B Methodol. 41, 202–217.

Zhang, X., Huang, C., Xu, Y., Xia, L., Dai, P., Bo, L., Zhang, J., Zheng, Y., 2021. Traffic flow forecasting with spatial-temporal graph diffusion network. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 15008–15015.

Zhang, S., Yao, Y., Hu, J., Zhao, Y., Li, S., Hu, J., 2019. Deep autoencoder neural networks for short-term traffic congestion prediction of transportation networks. Sensors 19 (10), 2229.