Multi-Graph Inductive Representation Learning for Large-Scale Urban Rail Demand Prediction under Disruptions

Dang Viet Anh Nguyen, J. Victor Flensburg, Fabrizio Cerreto, Bianca Pascariu, Paola Pellegrini, Carlos Lima Azevedo, Filipe Rodrigues

Multi-Graph Inductive Representation Learning for Large-Scale Urban Rail Demand Prediction under Disruptions

Dang Viet Anh Nguyen, J. Victor Flensburg, Fabrizio Cerreto, Bianca Pascariu, Paola Pellegrini, Carlos Lima Azevedo, Filipe Rodrigues

Multi-Graph Inductive Representation Learning for Large-Scale Urban Rail Demand Prediction under Disruptions

Dang Viet Anh Nguyen[1,*], J. Victor Flensburg[2], Fabrizio Cerreto[3], Bianca Pascariu[4], Paola Pellegrini[4], Carlos Lima Azevedo[1], Filipe Rodrigues[1]

[1]Department of Technology, Management, and Economics, Technical University of Denmark (DTU), 2800 Kongens Lyngby, Denmark
[2]Traffic Division, Banedanmark, 4100 Ringsted, Denmark
[3]Metroselskabet og Hovedstadens Letbane, 2300 København S, Denmark
[4]COSYS-ESTAS, Gustave Eiffel University, F-59650 Villeneuve d'Ascq, France

*Corresponding author: Dang Viet Anh Nguyen, email: andng@dtu.dk

Acknowledgements

This research was supported by Sortedmobility Grant agreement ID: 875022, funding from EU's Horizon 2020.

Author Contributions

Dang Viet Anh Nguyen: Conceptualization, Methodology, Writing - original draft, Formal analysis. J. Victor Flensburg: Conceptualization, Methodology, Writing - review and editing. Fabrizio Cerreto: Conceptualization, Methodology, Writing - review and editing. Bianca Pascariu: Conceptualization, Methodology, Writing - review and editing. Paola Pellegrini: Conceptualization, Methodology, Writing - review and editing. Carlos Lima Azevedo: Conceptualization, Methodology, Supervision, Validation, Funding acquisition, Writing - review and editing. Filipe Rodrigues: Conceptualization, Methodology, Supervision, Validation, Writing - review and editing. All authors reviewed the results and approved the final version of the manuscript.

Declaration of Conflicting Interests

The authors declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

March 5, 2026

Multi-Graph Inductive Representation Learning for Large-Scale Urban Rail Demand Prediction under Disruptions

Abstract

With the expansion of cities over time, Urban Rail Transit (URT) networks have also grown significantly. Accurate demand prediction plays a crucial role in supporting planning, scheduling, fleet management, and other operational decisions. This study proposes an Origin–Destination (OD) demand prediction model called Multi-Graph Inductive Representation Learning (mGraphSAGE) for large-scale URT networks under operational uncertainties. The proposed model represents each OD pair as a node in multiple graphs that capture distinct spatial and temporal correlations, thereby enhancing the spatial learning capability of graph-based methods while maintaining scalability. Moreover, operational uncertainties such as train delays and cancellations are explicitly incorporated as model inputs to improve robustness under real-world disruptions. The model is validated on three network scales of the Copenhagen URT system. Experimental results show that mGraphSAGE outperforms both conventional graph-based and machine learning baselines, achieving up to a 5% reduction in RMSE across network scales. The consistent improvement demonstrates the model's enhanced spatial representation and robustness under operational uncertainties, confirming its suitability for large-scale and disrupted URT environments.

Keywords:
OD demand prediction, Urban transit rail, Graph Neural Networks, Large scale

1. Introduction

In today's urban areas, the development of public transport is critical in addressing the challenges of traffic congestion, carbon emissions, and sustainable mobility. Along with bus systems, urban rail transit (URT) provides transportation for millions of daily trips in cities worldwide. For instance, the S-train system in Copenhagen, Denmark, served 106.179 million passengers in 2023 [1]. Therefore, maintaining a reliable and efficient URT system is essential not only for citizens' quality of life but also for economic productivity and environmental sustainability.

In the research field of URT operations, origin–destination (OD) prediction, estimating the number of passengers traveling between origin and destination stations, is a crucial task that supports planning, scheduling, fleet management, and other operational decisions [2]. Accurate OD demand prediction plays a vital role in enabling data-driven decision-making for public transport systems. Reliable forecasts allow operators to improve service reliability by proactively managing train schedules and capacity during disruptions, optimize operational efficiency through dynamic resource allocation, and enhance the passenger experience by reducing overcrowding and providing timely travel information. These practical implications

underscore the growing need for robust and resilient demand prediction models capable of operating effectively under both normal and disrupted conditions.

Unlike zone-based demand forecasting, which has been extensively studied in previous research, OD prediction is more challenging due to the complexities of URT network structures, transfer patterns, passenger travel behaviors, and partial observability [3]. With the rapid urbanization and expansion of URT networks in most cities worldwide, the scale of OD networks has increased significantly, raising the need for more scalable and accurate predictive models. Furthermore, passenger demand in URT systems is inherently influenced by the system's operational reliability; therefore, effective prediction models should explicitly account for reliability variations and adapt to dynamic changes in network conditions.

With the advancement of machine learning, computing power, and data availability, recent studies have explored advanced techniques in deep learning to capture complex dependencies and specific demand patterns in URT networks for OD prediction [2, 4, 5]. Most previous studies focus on tackling different aspects of OD demand prediction in URT networks, such as real-time data availability, sparsity, high dimensionality, and ridership quantity relationships. Very few studies address the challenge of predicting OD demand under disrupted scenarios including unexpected events like train cancellations or delays [6]. Additionally, although prediction models based on graph neural networks (GNNs) have been widely adopted in traffic prediction [7, 8, 9, 10], their application in URT OD demand prediction is still limited. GNNs are known to efficiently extract features with complex topological characteristics. However, one of the challenges related to their utilization is their scalability. Most traditional GNN models rely on spectral decomposition or matrix factorization techniques, which are transductive and do not scale well to large graphs.

Despite promising progress, existing OD demand prediction models still face several critical limitations. First, most conventional machine learning and deep learning approaches are designed for fixed network structures and thus struggle to generalize or scale to large and dynamically evolving URT networks. Second, while recent GNN-based models have shown strong potential for traffic and ridership prediction, they often rely on a single predefined spatial graph and do not explicitly distinguish between different types of spatial and temporal dependencies among OD pairs. Third, very few studies have incorporated the operational reliability of URT systems, such as train delays and cancellations, into the prediction process, even though these factors strongly influence passenger travel behavior. As a result, existing schemes are often less effective in handling real-world disruption scenarios and capturing multi-level correlations within the OD network. These gaps motivate our development of a scalable and robust multi-graph inductive representation learning framework for urban rail OD demand prediction.

Given these considerations, this study proposes a short-term OD demand prediction model that is both scalable for large-scale URT networks and robust under various scenarios of URT system reliability, based on graph inductive representation learning [11]. Specifically, the proposed method models the OD demand of the URT network as different graphs, establishing connectivity via various temporal and spatial correlations. The proposed graphs are then trained using graph inductive representation learning technique designed for inductively learning node representation in a large graph. By learning through sampling and aggregating features from neighboring ODs, we achieve a prediction model that performs well and scales effectively to large, complex real-world URT networks. Additionally, our model incorporates

unique features regarding URT system reliability, enhancing the prediction model's performance under train cancellation or delay scenarios. The main contributions of this paper are summarized as follows:

- We propose mGraphSAGE, an inductive OD-pair–level prediction framework for large-scale urban rail transit systems that represents each OD pair as a node and enables scalable learning under evolving network conditions.
- We introduce a multi-graph representation that captures complementary spatial and service-based dependencies among OD pairs, enabling scalable and expressive demand modeling.
- We incorporate operational reliability information, such as train delays and cancellations, to improve robustness of short-term OD demand prediction under disrupted operating conditions.
- Experiments on a real-world urban rail network demonstrate that the proposed approach outperforms existing machine learning baselines in both accuracy and robustness.

The remainder of this paper is organized as follows: Section 2 highlights previous related work on short-term OD demand prediction in URT. Section 3 details the research problem, graph modeling, feature extraction, and the proposed model - mGraphSAGE. Section 4 presents the computational experiments and comparative analysis with reference methods to validate the performance of the proposed model. Finally, Section 5 summarizes the main findings, limitations of the proposed model, and suggests possible future research directions.

2. Related Works

URT systems have been widely studied from various operational perspectives, including service reliability, passenger flow modeling, and disruption management. Previous research has examined the effects of train delays and cancellations on passenger behavior and network performance, aiming to enhance service recovery and system resilience [12, 13, 14]. However, much of this literature has focused on descriptive analyses or simulation-based evaluations, rather than developing predictive models capable of anticipating demand under operational uncertainties.

In parallel, a growing body of work has explored OD demand prediction using machine learning and deep learning methods to forecast passenger flows at different spatial and temporal scales. Despite notable progress, relatively few studies have focused on short-term URT OD demand prediction, primarily due to challenges that distinguish it from other transport prediction tasks. These challenges include limited real-time availability of OD data, severe data sparsity, and the high dimensionality of URT networks. Specifically, OD demand reflects individual passenger trips that can only be fully observed after trip completion [4, 5], leading to incomplete real-time information and reduced short-term forecasting accuracy [6]. Moreover, many OD pairs exhibit low or zero demand due to temporal variability and spatial heterogeneity [4], while the quadratic growth in the number of OD pairs with respect to network size results in extremely large and complex prediction spaces.

To address these challenges, recent studies have proposed various deep learning models that leverage advances in spatiotemporal representation learning. Jiang et al. [2] introduced a deep learning framework combining temporally shifted graph convolution and LSTM with an attention mechanism to capture spatial and temporal dependencies, together with a reconstruction module to mitigate partial observability of OD flows. Liu et al. [15] addressed real-time data availability by jointly learning from heterogeneous information sources, including incomplete OD matrices, unfinished order vectors, and destination-origin (DO) matrices. Zhang et al. [4] proposed a channel-wise attentive-split convolutional neural network with an inflow–outflow-gated mechanism to capture correlations between inflows, outflows, and OD flows. Noursalehi et al. [5] developed a multi-resolution spatiotemporal neural network that utilizes exit-based observations to handle partial observability, while Zhu et al. [16] proposed a two-stage OD prediction model that combines flow prediction and separation rate estimation. Cheng et al. [17] employed singular value decomposition and low-rank high-order vector autoregression to approximate OD data and address availability issues through time-evolving features. More recently, Zhang et al. [6] proposed a physics-guided adaptive graph spatiotemporal attention network with a masked loss function, together with real-time OD estimation and matrix compression techniques to alleviate sparsity and incomplete observability. Dong et al. [18] developed a deep learning-based method for short-term passenger flow forecasting in urban rail systems, highlighting the ability of neural architectures to capture complex demand fluctuations under real operational conditions.

Beyond modeling accuracy, increasing attention has been paid to scalability and robustness in large-scale URT OD prediction. Xing et al. [19] proposed a multi-graph spatiotemporal fusion framework that integrates multiple station-level relational graphs to enhance OD forecasting performance. From an operational analytics perspective, Wang et al. [20] introduced an ensemble deep learning architecture with data augmentation to improve robustness for medium- and low-flow OD pairs at the network level, while Duan et al. [21] proposed a multi-resolution spatiotemporal semantic learning approach that mitigates high-dimensional sparsity through hierarchical spatial representations and ConvLSTM-based temporal modeling. In related domains, recent studies have also explored the integration of multi-graph neural networks and resource-aware architectures to improve prediction performance and computational efficiency in large-scale traffic systems [22, 23]. Although these works reflect a growing interest in scalable and energy-efficient graph learning frameworks, their applications have largely focused on road traffic flow rather than urban rail demand prediction.

More recently, disruption-aware OD prediction frameworks have emerged to explicitly model passenger flow dynamics during unexpected events. For example, Zou et al. [24] proposed a real-time OD prediction framework that integrates incident matrices with enhanced graph convolution operations to capture abnormal flow variations during underground incidents. Similarly, Fan et al. [25] developed a deep counterfactual inference framework with dual-channel learning to estimate urban rail transit OD matrices under operational disruptions, enabling explicit comparison between normal and incident-affected scenarios.

While these studies clearly demonstrate the importance of explicitly modeling operational disruptions, they predominantly rely on transductive OD-matrix–based formulations defined over fixed sets of OD pairs. Such formulations can limit scalability and adaptability in large-scale or dynamically evolving URT networks, where the number of OD pairs grows rapidly and network topology may change over time. In contrast, our work focuses on an inductive

graph representation learning framework that enables scalable OD demand prediction while maintaining robustness under operational disruptions.

Despite these advances, several important gaps remain. First, most existing approaches are station-centric or OD-matrix-based, which constrains scalability and limits generalization to large URT networks with thousands of OD pairs. Second, while data sparsity and partial observability have been extensively studied, relatively few models explicitly integrate operational disruptions, such as train delays and cancellations, within an inductive predictive framework. Third, the majority of current methods are transductive and require retraining when the network topology or OD set changes, which is impractical for real-world urban rail operations.

To address these limitations, this study proposes mGraphSAGE, an inductive multi-graph learning framework for large-scale OD demand prediction in URT systems. By modeling each OD pair as a node and learning representations across multiple complementary graphs, the proposed approach efficiently handles data sparsity and incomplete OD information, generalizes across different network configurations without retraining, and maintains robust performance under diverse operational scenarios, including disruptions. By incorporating operational uncertainties into a scalable graph-based learning paradigm, this research contributes toward improving the resilience and reliability of URT demand prediction.

3. Methodology

In this section, we present our research problem on OD prediction in URT, describe the features in our data, and introduce the proposed mGraphSAGE model for OD prediction in URT systems.

3.1. Research problem and features

The OD demand prediction task at the network level is challenging due to the complex spatial and temporal dependencies among the demands of different OD pairs. Additionally, the level of observability of passenger trips in the metro at the predicted time step varies because each OD demand can only be recorded after the passenger has completed their trip. This makes some of the OD demand only partially observed at prediction time. To structure the prediction timeline, we divide the prediction horizon into discrete intervals; at the beginning of each interval, the prediction task utilizes historical information to forecast demand for that specific interval.

Let d_t^i be the number of passengers departing at time interval t and having already completed their trip through OD $i \in V$ (the set of OD pairs) before the prediction time. p_t^o denotes the number of passengers departing at origin station o of OD i but not yet completed their trip at prediction time. Let $x_t^i = d_t^i + p_t^o$ be the sum of the number of trips completed through OD i and the number of uncompleted trips departing from the origin station of OD i. Thus, $\mathbf{D}_t = \{d_t^1, d_t^2, \ldots, d_t^n\}$ is the vector of completed trips of all OD pairs on the graph departing at time interval t. $\mathbf{P}_t = \{p_t^1, p_t^2, \ldots, p_t^n\}$ is the vector of uncompleted trips of all n origin stations of n ODs. $\mathbf{X}_t = \{x_t^1, x_t^2, \ldots, x_t^n\}$ is the vector of aggregated passenger demand traveling through all OD pairs (the sum of completed trips on an OD pair and passengers departing from the origin station of the respective OD pair) departing at time interval t.

Figure 1 illustrates how completed trips $\mathbf{D}$, uncompleted trips $\mathbf{P}$, and aggregated demand $\mathbf{X}$ are extracted for each prediction time and for a given OD. For instance, if the prediction time is 8:40, in the interval $t-1$ from 8:00 to 8:20, there are a total of 4 passengers departing, but only 2 of them completed their trip before 8:40. Therefore, we identify $x_{t-1} = 4$, $p_{t-1} = 2$, and $d_{t-1} = 2$. Another thing to note is that x_t, p_t, and d_t will be updated as the prediction time progresses. For example, when the prediction time moves to 9:00, and the trip of passenger 4 is completed at 8:50, then $d_{t-1} = 3$ and $p_{t-1} = 1$ instead of 2 and 2 as previously.Using the information of aggregated demand on each OD, $\mathbf{X}$, and completed trips on each OD, $\mathbf{D}$, we aim to tackle the partial observability challenges in the OD prediction task.

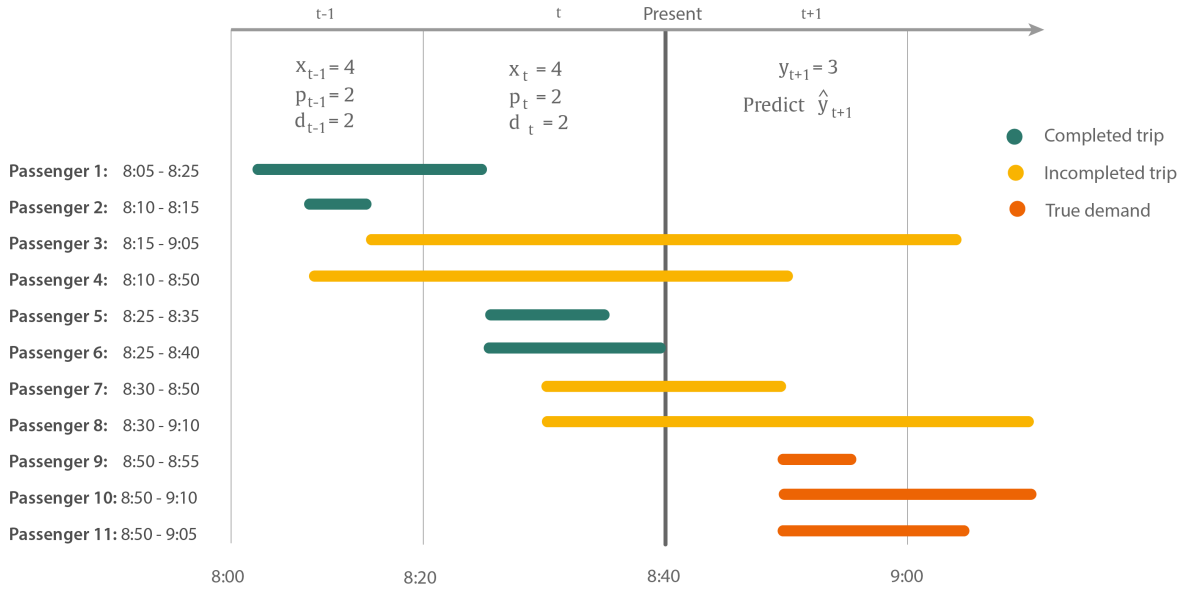


Figure 1: Visualization of demand feature extraction for a given OD and for a given prediction time
This example illustrates the temporal evolution of demand-related features for one station involved in an OD pair. x_t, p_t, and d_t denote the aggregated demand, uncompleted trips, and completed trips, respectively, at time interval t. Similarly, $\hat{y}_{t+1}$ and y_{t+1} represent the predicted and true demand at time interval $t+1$. The values of x_t, p_t, and d_t are updated dynamically as time progresses.

According to [26, 9], OD demand temporal dependencies have two main types: *tendency* and *periodicity*. The former refers to present demand being affected by previous demand in the last several time intervals, while the latter relates to the repetition of daily and weekly demand patterns. To account for *tendency*, we gather the data of $\mathbf{X}$ and $\mathbf{D}$ from the last 8 time intervals before the prediction time, specifically $\{\mathbf{X}_t, \mathbf{X}_{t-1}, \ldots, \mathbf{X}_{t-7}\}$ and $\{\mathbf{D}_t, \mathbf{D}_{t-1}, \ldots, \mathbf{D}_{t-7}\}$. The choice of the number of previous intervals needs to be determined by empirical research. To capture daily and weekly patterns from *periodicity*, we use the following features:

- One-hot encoding features for indicating day of the week: $\mathbf{f}_w$;
- One-hot encoding features for indicating time interval of the day: $\mathbf{f}_t$.

We also add one-hot encoded node ID features $\mathbf{f}_{id}$ to distinguish different nodes (OD pairs) on the graph. However, in some case using one-hot encoding for each OD pair in the large-scale case would result in a massive increase in feature dimensions. To avoid this, we use a different encoding method to distinguish OD pairs. We add a node-ID feature as a one-dimensional vector with a size equal to the number of stations. The element having index corresponding to the origin station is indicated as 1, while the element having index corresponding to the destination station is indicated as -1.

On top of *tendency* and *periodicity*, we capture *reliability* to account for impact of operations reliability of the URT system in demand. Therefore, we incorporate 12 supply features denoted by $\mathbf{f}_s$, which measure the system's reliability for each OD pair. According to [27, 28], popular reliability measures of public transportation systems include waiting time, regularity, and travel time. In the context of our study, regularity refers to the frequency of train arrivals and cancellations within one hour, while waiting time and travel time cover the delays of trains at specific stations relative to the OD pairs. Therefore, we measure the reliability of the system for each OD pair based on three dimensions at both the origin and destination stations as follows:

- The number of trains at the station during the last hour for each line, with respect to the OD direction;
- Mean delay of trains at the station during the last hour for each line, with respect to the OD direction;
- Proportion of trains that have been cancelled at the station during the last hour for each line, with respect to the OD direction.

Because each station has multiple lines, we take the mean and maximum values for each dimension at both the origin and destination stations. Therefore, we have a total of 12 features related to the reliability of the railway system.

Given all the observed aggregated OD demand data and the completed trips from all ODs and additional information such as date, time, and system reliability up to time interval t, our study aims to learn a function $f(\cdot) : \mathbb{R}^{N \times F} \rightarrow \mathbb{R}^N$ to predict the complete OD demand of the urban rail network for the next time interval $\hat{\mathbf{y}}_{t+1}$:

$$[\mathbf{X}_{t-7}, \ldots, \mathbf{X}_t, \mathbf{D}_{t-7}, \ldots, \mathbf{D}_t, \mathbf{f}_w, \mathbf{f}_t, \mathbf{f}_{id}, \mathbf{f}_s] \xrightarrow{f(\cdot)} \hat{Y}_{t+1}$$

Table 1 summarizes the key notations used throughout the paper.

3.2. OD graph in URT

In conventional modeling of the URT network as a graph, stations are represented as vertices, with edges indicating passenger flow between them. While this approach provides a clear and intuitive understanding of the network structure, it does not effectively represent the relationships in demand patterns between OD pairs and can lead to significant graph size increases when considering OD demand as edges. In this study, we address the OD prediction problem in a large-scale URT network by modeling each OD pair as a vertex in the graph. This strategy reduces both the complexity and size of the graph, facilitating more efficient

Table 1: Summary of Notations

Symbol	Description
Graph and Data Representation	
$G_k = (V, E_k)$	The k-th graph representing OD pair relationships
K	Total number of graphs used in the multi-graph framework
V	Set of OD nodes, where each node represents an OD pair
E_k	Set of edges in the k-th graph
$N = \|V\|$	Total number of OD nodes
$\mathbf{A}_k \in \{0,1\}^{N \times N}$	Adjacency matrix of the k-th graph
D_{ij}^k	Distance between two node $i, j \in V \quad \forall i \neq j$ in graph k
$\mathbf{X} \in \mathbb{R}^{N \times T}$	Aggregated OD demand; each row corresponds to one OD pair
$\mathbf{D} \in \mathbb{R}^{N \times T}$	Completed trips departing from the origin station of OD $i \in V$
$\mathbf{P} \in \mathbb{R}^{N \times T}$	Uncompleted trips departing from the origin station of OD $i \in V$
$\mathbf{f}_w$	Day indicator features
$\mathbf{f}_t$	Time interval indicator features
$\mathbf{f}_i d$	Node ID features
F	Number of input features per OD pair
$Y \in \mathbb{R}^N$	True OD demand vector at the prediction horizon
$\hat{Y} \in \mathbb{R}^N$	Predicted OD demand vector
Graph Neural Network Representation Learning	
$\mathbf{h}_v^{(l)}$	Hidden representation of node v at GNN layer l
$\mathbf{H}^{(l)} \in \mathbb{R}^{N \times d_l}$	Node embedding matrix at layer l with dimension d_l
$\mathbf{W}^{(l)} \in \mathbb{R}^{d_{l-1} \times d_l}$	Trainable weight matrix at layer l
$\mathcal{N}_v^{(k)}$	Set of neighbors of node v in the k-th graph
AGGREGATE$(\cdot)$	Neighborhood feature aggregation function
MEAN$(\cdot)$	Mean aggregation function
CONCAT$(\cdot)$	Concatenation of arrays
$\sigma(\cdot)$	Nonlinear activation function
Training and Optimization	
$f(\cdot)$	Function that maps input features to prediction outputs
$\mathcal{L}$	Loss function
Θ	Set of all trainable model parameters
η	Learning rate of the optimizer
E	Number of training epochs
B	Batch size during training

node prediction for OD demand. Specifically, we define an OD graph $G = (V, E, A)$, where V is the set of vertices representing OD pairs and E represents the relationship in demand patterns between these pairs. The adjacency matrix $A \in \mathbb{R}^{n \times n}$ defines the relationships between the vertices, where n is the number of vertices (OD pairs) and the element A_{ij} indicates the presence of an edge between vertices i and $j \in V$.

3.3. The mGraphSAGE model

In this section, we detail the proposed method based on multi-graph inductive representation learning (mGraphSAGE) for predicting OD demand in a large scale URT network.

3.3.1. Modeling temporal-spatial correlations between OD pairs

A key challenge in applying GNN-based methods for OD demand prediction is defining how the nodes, representing OD pairs, are connected. In an urban rail network, OD pairs may exhibit dependencies not only through geographical proximity but also through the similarity of their temporal demand dynamics. Therefore, the way edges are defined has a direct impact on how effectively the model captures these spatio-temporal correlations.

To comprehensively represent both aspects, we design four complementary graphs, each emphasizing a distinct type of relationship among OD pairs: (i) a temporal correlation-based graph $G_t(V, E, A^t)$ that encodes dynamic similarity in travel demand, (ii) centroid distance graph $G_s(V, E, A^s)$ that captures overall spatial proximity, (iii-iv) origin and destination distance graphs, $G_o(V, E, A^o)$ and $G_d(V, E, A^d)$ that capture localized spatial correlations around specific origin and destination areas. Together, these graphs allow mGraphSAGE to jointly learn how OD demand evolves across time and propagates across space at different levels of granularity.

We adopt distance-based approaches rather than clustering-based measures because they offer a more interpretable and fine-grained way to evaluate pairwise relationships between OD pairs. Distance metrics enable transparent control over connectivity (via thresholds) and preserve subtle but important relational variations that clustering may overlook.
Following are the details and motivations of the four graphs we propose.

(i) Temporal correlation-based graph. The temporal graph G_t aims to connect OD pairs that show similar temporal demand evolution. The intuition is that even geographically distant OD pairs can have correlated travel patterns due to shared external factors, such as working-hour peaks, special events, or disruptions in nearby lines. We use Dynamic Time Warping (DTW) to quantify the similarity between the historical demand profiles of OD pairs during the previous interval t, forming the temporal distance matrix D^t. A threshold σ_t determines whether a pair of OD nodes should be linked:

$$A_{ij}^t = \begin{cases} 1, & \text{if } D_{ij}^t \leq \sigma_t \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

This enables the model to share information among OD pairs exhibiting synchronized or lagged temporal demand patterns.

(ii) Centroid distance graph. While the temporal graph captures dynamic similarity, the centroid distance graph G_s focuses on spatial proximity. The idea is that OD pairs located close to each other in space tend to reflect similar travel behaviors and demand magnitudes because they share overlapping service areas and accessibility zones. Following [9], we measure the Euclidean distance between OD centroids:

$$D_{ij}^s = \sqrt{\left(\frac{r_j^o + r_j^d}{2} - \frac{r_i^o + r_i^d}{2}\right)^2 + \left(\frac{s_j^o + s_j^d}{2} - \frac{s_i^o + s_i^d}{2}\right)^2} \quad \forall i, j \in V \tag{2}$$

where $r_i^o, s_i^o, r_i^d, s_i^d$ are the UTM coordinates of the origin and destination stations of OD pair i, and $r_j^o, s_j^o, r_j^d, s_j^d$ are those of OD pair j. Two OD pairs are connected ($A_{ij}^s = 1$) if their centroid distance D_{ij}^s is smaller than a threshold σ_s. This graph captures overall geographical correlations between travel flows.

(iii) Origin distance graph and (iv) destination distance graph. OD pairs may also exhibit localized spatial dependencies on either the origin or destination side. For example, OD pairs departing from nearby stations may respond similarly to local congestion or station-specific events, even if their destinations differ. Likewise, OD pairs terminating at nearby destinations often reflect overlapping passenger arrival patterns.

To represent these fine-grained spatial relationships, we define two additional graphs: the origin distance graph G_o and the destination distance graph G_d, whose distances are computed as:

$$D_{ij}^o = \sqrt{(r_j^o - r_i^o)^2 + (s_j^o - s_i^o)^2}, \quad \forall i, j \in V \tag{3}$$

$$D_{ij}^d = \sqrt{(r_j^d - r_i^d)^2 + (s_j^d - s_i^d)^2}, \quad \forall i, j \in V \tag{4}$$

Using thresholds σ^o and σ^d, we construct the adjacency matrices A^o and A^d. These two graphs enrich spatial modeling by distinguishing departure-side and arrival-side dependencies among OD pairs information that a single global spatial graph might miss.

3.3.2. Multi-graph inductive representation learning (mGraphSAGE)

Figure 2 illustrates the architecture of mGraphSAGE, where F, N, and B denote the feature length, the number of nodes (OD pairs), and the batch size, respectively. We use four adjacency matrices A^t, A^s, A^o, A^d (as discussed in Section 3.3.1) to establish the connections of the graph, forming four graphs G_t, G_s, G_o, and G_d. These four graphs are then trained in four separate channels using the GraphSAGE algorithm for learning node representations.

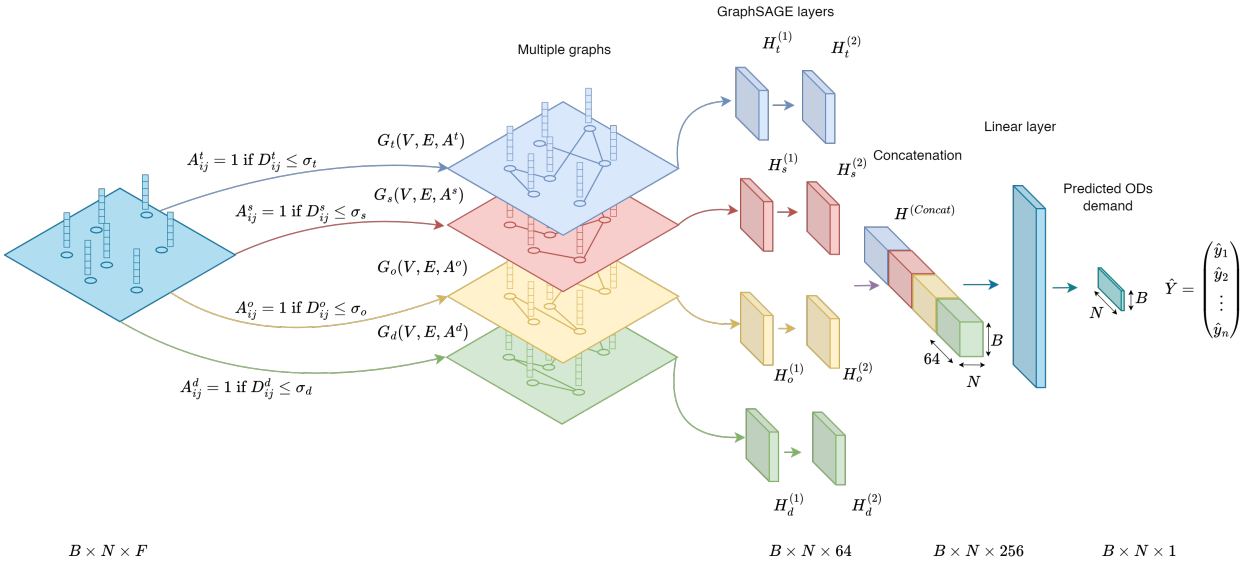


Figure 2: Overview of the mGraphSAGE framework.
The model independently processes four relational graphs (G_t, G_s, G_o, G_d) representing temporal and spatial dependencies, and then aggregates the resulting embeddings into a unified representation.

GraphSAGE (Graph Sample and Aggregation) is a method for inductively learning node embeddings in large graphs. First introduced by Hamilton et al. [11], this method addresses scalability and generalization issues in traditional graph convolutional networks (GCNs). GraphSAGE is the first inductive learning method that, instead of learning specific embeddings for nodes, learns an aggregation function that can be applied to any node's neighborhood, regardless of whether the node was present during training. This property allows GraphSAGE to generalize to unseen nodes or graphs without the need for retraining. As a result, a prediction model based on GraphSAGE is an efficient way to handle the sparsity in OD demand data within URT networks.

Given a graph $G = (V, E)$, where V is the set of nodes and E is the set of edges, let $\mathbf{h}_v^{(l)}$ denote the embedding of node v at the l-th layer of the model. The GraphSAGE node embedding generation procedure is as follows:

1. GraphSAGE samples a fixed set of neighbors $\mathcal{N}(v)$ for each node v, rather than using all neighbors. This sampling helps to efficiently handle large graphs. Once neighbors $\mathcal{N}(v)$ are sampled, GraphSAGE uses an aggregation function AGGREGATE_l at layer l to combine information from these neighboring nodes:
$$\mathbf{h}_{\mathcal{N}(v)}^{(l)} = \text{AGGREGATE}_l\left(\left\{\mathbf{h}_u^{(l-1)} \mid u \in \mathcal{N}(v)\right\}\right),$$
where $\mathbf{h}_u^{(l-1)}$ is the embedding of node u at layer $l-1$.
2. The aggregated information $\mathbf{h}_{\mathcal{N}(v)}^{(l)}$ is then used to update the representation of each node $\mathbf{h}_v^{(l)}$. This step, known as node embedding update, combines both the node's own features and the aggregated features from its neighbors. This is done using a fully connected layer with a nonlinear activation function σ:
$$\mathbf{h}_v^{(l)} = \sigma\left(\mathbf{W}^{(l)} \cdot \text{CONCAT}\left(\mathbf{h}_v^{(l-1)}, \mathbf{h}_{\mathcal{N}(v)}^{(l)}\right)\right).$$

Here, $\mathbf{W}^{(l)}$ denotes the weight matrix at layer l, and CONCAT$(\cdot)$ denotes the concatenation operation along specified dimensions.

3. Different types of aggregation functions can be used, such as mean aggregator, LSTM aggregator, and pooling aggregator. In this study, we use the mean aggregator. Thus, the node representation update using the mean aggregator is given by:

$$\mathbf{h}_v^{(l)} \leftarrow \sigma\left(\mathbf{W}^{(l)} \cdot \text{MEAN}\left(\left\{\mathbf{h}_v^{(l-1)}\right\} \cup \left\{\mathbf{h}_u^{(l-1)} \mid u \in \mathcal{N}(v)\right\}\right)\right),$$

where MEAN$(\cdot)$ denotes the mean aggregation function applied to the concatenated embeddings of node v and its neighbors.

GraphSAGE performs these processes iteratively across multiple layers (or hops). In each layer, nodes aggregate information from neighbors sampled in the previous layer, allowing the model to capture information from increasingly distant parts of the graph. This process enables GraphSAGE to effectively learn node embeddings that encode both local and broader structural information from the graph.

In our case, we use two SAGEConv layers for training each graph so that each node in the graph can aggregate information not just from immediate neighbors (1-hop neighbors) but also from neighbors of its neighbors (2-hop neighbors), thus incorporating more contextual information from the graph. The ReLU$(\cdot)$ activation function applied after the first SAGEConv layer introduces non-linearity, allowing the network to learn more complex mappings. The output node representations of the four graphs are then concatenated before applying the ReLU activation function and feeding into a linear layer to get the demand prediction for each OD pair on the graph.

Let G_i denote the i-th graph where $i \in \{t, s, o, d\}$ and F_i be the input node features for graph G_i. The calculation procedure is shown in Equations 5-10.

$$H_i^{(1)} = \text{SAGEConv}_1(F_i, A^i), \quad \text{for } i \in \{t, s, o, d\} \tag{5}$$

$$H_i^{(1,\text{ReLU})} = \text{ReLU}(H_i^{(1)}), \quad \text{for } i \in \{t, s, o, d\} \tag{6}$$

$$H_i^{(2)} = \text{SAGEConv}_2(H_i^{(1,\text{ReLU})}, A_i), \quad \text{for } i \in \{t, s, o, d\} \tag{7}$$

$$H^{(\text{Concat})} = \text{Concat}(H_t^{(2)}, H_s^{(2)}, H_o^{(2)}, H_d^{(2)}) \tag{8}$$

$$H^{(\text{Concat,ReLU})} = \text{ReLU}(H^{(\text{Concat})}) \tag{9}$$

$$\hat{Y} = \text{Linear}(H^{(\text{Concat,ReLU})}) \tag{10}$$

Here, H_i represent the feature representation of nodes in graph G_i and $\hat{Y}$ is the predicted demand for all ODs in urban rail.

In our framework, the temporal and spatial relationships are represented by separate graphs, each with its own adjacency matrix (A^t, A^s, A^o, A^d). These graphs are not multiplexed into a single layer; instead, mGraphSAGE processes them independently and subsequently integrates the learned node embeddings across graph types through aggregation. Specifically, G_t encodes temporal correlations based on similarity in OD demand evolution, while G_s, G_o, and G_d represent spatial proximities at different levels (overall, origin-side, and destination-side). This design enables the model to flexibly learn from heterogeneous

relational structures without conflating them into a single multiplex adjacency, thereby preserving the interpretability and modularity of each relational dimension.

The overall training procedure of mGraphSAGE is presented in Algorithm 1.

Algorithm 1: mGraphSAGE for OD Demand Prediction in Urban Rail Networks

Input: OD demand data $\mathcal{D}$;
Station coordinates $\mathcal{C}$;
Temporal features $\mathbf{f_w}, \mathbf{f_t}$;
Node (OD) ID features $\mathbf{f_{id}}$
Supply features $\mathbf{f_s}$;
Hyperparameters: number of layers L, embedding dimension d, learning rate η.
Output: Predicted OD demand $\hat{\mathbf{y}}$ for future time steps.

1. Preprocessing:
Compute node features $[\mathbf{X}_{t-7}, \ldots, \mathbf{X}_t, \mathbf{D}_{t-7}, \ldots, \mathbf{D}_t, \mathbf{f}_w, \mathbf{f}_t, \mathbf{f}_{id}, \mathbf{f}_s]$
Normalize and align OD data across all time steps.

2. Multi-Graph Construction:
Build temporal correlation-based graph $G_t = (V, E_t)$ using Equation 1
Build centroid distance graph $G_s = (V, E_s)$ using Equation 2
Build origin distance graph and destination distance graph $G_o = (V, E_o)$ and $G_d = (V, E_d)$ using Equations 3 and 4

3. GraphSAGE Representation Learning:
for *each graph* G_i *with* $i \in t, s, o, d$ **do**
  for *each layer* $l = 1, \ldots, L$ **do**
    for *each node* $v \in V$ **do**
      Aggregate neighbor features:
      $\mathbf{h}_{\mathcal{N}(v)}^{(l)} = \text{AGGREGATE}_l\left(\left\{\mathbf{h}_u^{(l-1)} \mid u \in \mathcal{N}(v)\right\}\right)$
      Update node embedding:
      $\mathbf{h}_v^{(l)} \leftarrow \sigma\left(\mathbf{W}^{(l)} \cdot \text{MEAN}\left(\left\{\mathbf{h}_v^{(l-1)}\right\} \cup \left\{\mathbf{h}_u^{(l-1)} \mid u \in \mathcal{N}(v)\right\}\right)\right)$
Concatenate embeddings from multiple graphs:
$\mathbf{h}^{(\text{Concat})} = \text{Concat}(\mathbf{h}_t^{(L)}, \mathbf{h}_s^{(L)}, \mathbf{h}_o^{(L)}, \mathbf{h}_d^{(L)})$

4. Prediction and Training:
Predict OD demand: $\hat{Y}_v = \text{MLP}(\mathbf{h}^{(\text{Concat})})$
Compute loss: $\mathcal{L} = \frac{1}{|V|}\sum_v (\hat{Y}_v - Y_v)^2$
Update parameters θ via gradient descent with learning rate η.

5. Output:
Return predicted OD demand $\hat{Y}$ for future time steps.

4. Computational Experiments

In this section, we present our experiments conducted for evaluating the proposed method with a real-world case study on Copenhagen S-train URT.

4.1. Data description

We introduce a real-world case study for short-term OD prediction in URT based on data from the Copenhagen suburban railway, known as the S-train. The railway network

consists of 170 km of electrified double track with homogeneous passenger traffic. The OD-demand dataset is constructed based on the Danish nationwide Automatic Fare Collection (AFC) system called "Rejsekort", a smart card system where users tap in (origin) and tap out (destination) are recorded. We collect demand information over 84 stations in the railway network over an 11-month period from January 29, 2021, to December 3, 2021. Data are collected only 5 days per week during peak periods from 5 AM to 12 PM. We adjust for holidays by using the historical average demand from the week before each holiday. The final demand data encompass tap-in and tap-out at all stations in the rail network (84 stations), which are then used to prepare the aggregated demand feature $\mathbf{X}$ and completed trips of all ODs $\mathbf{D}$ with a time interval of $t = 20$ minutes. We note that this AFC dataset does not account for all demand in the network, but it represents the main demand patterns on the Danish public transportation network [12]. Future work is required to include additional data to scale up the current framework to the total demand.

In addition to the tap-in and tap-out demand data, we also collect information on the number of trains, train delay times, and the number of trains canceled at each station per hour for each line and direction, which are then used to construct the supply feature vector $\mathbf{f}_s$. The geographic coordinates of the 84 stations are obtained via the open API of Rejseplanen, a public transport information platform in Denmark.

To ensure that hyperparameter tuning does not exploit test information, model development and parameter optimization are conducted entirely on a separate dataset covering 52 weeks between January 29, 2017 and January 27, 2018. During model development, 20% of this dataset is reserved as a validation set to guide hyperparameter selection, with early stopping applied based on validation performance. The 2021 dataset is subsequently used only once for final training and evaluation, ensuring a fully unseen testing scenario and preventing data leakage.

4.2. Experimental setup and baseline methods

We evaluate our proposed method in two dimensions: scalability and robustness. For the scalability dimension, we test our method on three different graph scales of the railway network, including the "12 ODs" case, the "Tiny Copenhagen" case, and the "Full Copenhagen" case. In the 12 ODs case, we select the 12 OD pairs with the highest average demand. The "Tiny Copenhagen" case encompasses 12 contiguous stations in the network, resulting in 132 OD pairs, while the "Full Copenhagen" case covers the entire railway network with 84 stations, resulting in 6972 OD pairs (see Figure 3).

(a) Tiny Copenhagen case

(b) Full Copenhagen case

Figure 3: URT line maps of real-world S-train case studies
Source: DSB S-train official website (`https://www.dsb.dk/s-tog/`.)

The proposed mGraphSAGE model is implemented using PyTorch Geometric. For each OD graph (temporal, spatial, origin-based, and destination-based), the model employs two stacked GraphSAGE layers with 64 hidden units, ReLU activation, and a dropout rate of 0.5. The latent representations from the four graph branches are concatenated and passed through a fully connected regression layer to predict the next-interval OD demand. The model is trained using mean squared error (MSE) loss. Dynamic features include an 8-lag history of OD demand and completed trips, supplemented with date-time features, node identifiers, and operational supply variables such as train delays and cancellations.

Model development and hyperparameter tuning are performed entirely on a separate dataset covering the period from January 2017 to January 2018. A validation subset, comprising 20% of this dataset, is used for early stopping and overfitting prevention. The 2021 dataset is excluded from the model development process and is used only once for the final evaluation to assess generalization performance. We adopt Optuna [29] for automatic hyperparameter optimization. The hyperparameter search space and the final selected values are summarized in Table 2. During hyperparameter tuning, early stopping is applied if the validation loss does not improve for 100 consecutive training epochs.

Table 2: Hyperparameter search space and final selected values for mGraphSAGE

Hyperparameter	Range evaluated	12 ODs	Tiny CPH	Full CPH
Hidden dimension	{32, 64, 128}	128	64	64
Learning rate	$[5e-4, 1e-6]$	2e-4	1e-4	1e-6
Dropout	$[0.2, 0.6]$	0.6	0.5	0.5
Batch size	{32, 64, 128}	128	128	128
Neighbor sample	{5, 10, 15, 20, 25}	5	10	15
Maximum epochs	{2,000–15,000}	5,000	5,000	5,000
Optimizer	Adam	Adam	Adam	Adam
Loss function	MSE	MSE	MSE	MSE
Weight decay	$[1e-3, 1e-6]$	3e-5	2e-4	1e-6

For comparative analysis, we employ different prediction methods as baselines as follows:

- **RR**: Ridge Regression is a type of linear regression that includes a regularization term to prevent overfitting, especially when dealing with multicollinearity or when the number of predictors is larger than the number of observations [30]. The regularization parameter λ is tuned from 1e-3 to 1e-7, with the selected value being 1e-6.

- **XGB**: XGBoost is an advanced implementation of gradient-boosted decision trees designed to be highly efficient, flexible, and portable [31]. XGBoost provides a significant improvement over traditional gradient-boosting algorithms for different classification and regression predictive modeling problems. The learning rate is tuned among 0.1, 0.01, and 0.001, while the maximum depth of the tree is tuned to 2, 4, and 6. For the 12 ODs case, the selected learning rate is 0.1, while for the Tiny and Full Copenhagen cases, the selected learning rate is 0.001. The best parameter for maximum depth is 6.

- **GCN**: The Graph Convolutional Network (GCN) is adopted as a baseline graph representation learning method and serves as the foundation for many OD demand prediction models. The model architecture consists of two graph convolutional layers with 128 hidden units, each followed by ReLU activation and dropout (0.5), and a final linear layer for regression. Unlike mGraphSAGE, the GCN baseline performs full-batch message passing on the temporal graph, making it computationally more intensive and less scalable for large network cases. The hyperparameter tuning procedure for the GCN followed the same process as for mGraphSAGE, using Optuna over the same search ranges specified in Table 2. The selected learning rates for the 12-OD case, Tiny Copenhagen, and Full Copenhagen were 2e-4, 1e-6, 1e-6 respectively, using the MSE loss.

- **STGCN**: The Spatio-Temporal Graph Convolutional Network (STGCN) [32] is adopted as a representative spatiotemporal graph learning baseline that jointly models spatial dependencies through graph convolutions and temporal dynamics using temporal convolutional layers. The STGCN architecture consists of two spatio-temporal convolutional (ST-Conv) blocks, each containing a temporal gated convolution layer, a graph convolution layer, and another temporal gated convolution layer, followed by a

fully connected output layer. Each graph convolution layer utilizes 64 hidden channels, and the temporal convolution kernel size is set to 3. Residual connections and batch normalization are applied within each ST-Conv block to stabilize training. For the OD prediction task, STGCN represents OD pairs as nodes, using similar features to mGraphSAGE (i.e., historical OD lag features, datetime, and supply features) as node features. Since explicit physical connectivity between OD pairs is not well-defined, the adjacency matrix is constructed using a temporal correlation-based graph derived from DTW distances between historical OD flow time series. STGCN is trained with the Adam optimizer, a learning rate of 10^{-6} and MSE loss. Hyperparameters are selected based on the original STGCN design and commonly adopted settings in traffic forecasting studies.

To address the Full Copenhagen case and manage a large-scale graph within a reasonable computing and training time, we propose several alternatives. Specifically, to formulate a temporal correlation-based graph, we use the Fast Fourier Transform (FFT) instead of DTW to measure the similarity of demand time series data between OD pairs. DTW has a time complexity of $O(n^2)$, which is significantly more time-consuming compared to FFT's $O(n \log n)$. For the origin-destination distance graph, instead of setting a threshold to establish connections between OD pairs, we limit the number of edges to 10,000. This limit is determined empirically through experiments to ensure an appropriate training time. This helps avoid increasing graph complexity, which could slow down the training process.

We evaluate the prediction error of all models using two metrics: RMSE (Root Mean Square Error) and MAE (Mean Absolute Error). To assess the model's performance under different reliability conditions of the URT system, we test the prediction model across periods with varying levels of train delays and cancellations, as follows:

- All periods: the entire test time horizon.
- Periods with an average number of train cancellations at the origin or destination station in the last hour greater than 0.
- Periods with an average train delay time at the origin or destination station in the last hour greater than 60, 180, and 300 seconds.

All models are implemented and executed on a desktop workstation equipped with an AMD Ryzen Threadripper 3960X 3.8 GHz CPU, 128 GB of RAM, and an NVIDIA GeForce RTX 3080 Ti GPU. Table 3 summarizes the training time and peak GPU memory usage of GCN, STGCN, and mGraphSAGE across three network scales.

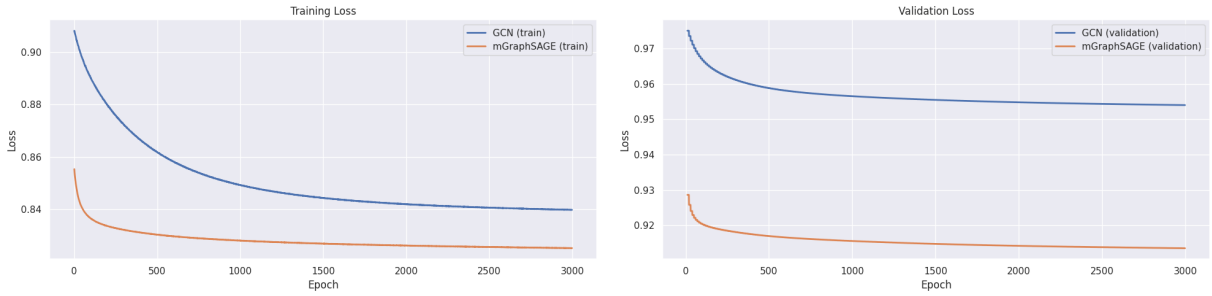
For the small-scale 12-OD case, STGCN converges rapidly, requiring only 2 minutes of training time. However, it incurs substantially higher GPU memory consumption compared to both GCN and mGraphSAGE (1764.2 MB compared to 112.1 MB for mGraphSAGE and 122.7 MB for GCN). This behavior stems from the inherent computational complexity of STGCN's architecture, which applies dense spatiotemporal convolutions across all nodes and time steps. Specifically, STGCN maintains high-dimensional intermediate feature maps for each node over the entire temporal window and performs full-graph convolutions at every temporal layer, resulting in memory and computational costs that scale with the number of nodes, temporal window length, and hidden feature dimensions.

As the network scale increases, this complexity leads to rapid growth in memory usage. In the Tiny Copenhagen case, STGCN's peak GPU memory usage exceeds 10850.6 MB, and in the Full Copenhagen network, it becomes infeasible to train within the available GPU memory. These results highlight the limited scalability of STGCN for large-scale OD prediction tasks, especially when dense OD graphs and long temporal histories are considered.

In contrast, both GCN and mGraphSAGE exhibit more favorable scaling behavior. mGraphSAGE consistently achieves comparable or lower peak GPU memory usage than GCN while also converging faster, especially in the larger Tiny Copenhagen and Full Copenhagen cases. This advantage is primarily due to its inductive neighborhood sampling strategy, which avoids full-graph spatiotemporal convolutions and limits both memory usage and computational cost by operating on localized neighborhoods. The scalability benefit of mGraphSAGE is most evident in the Full Copenhagen network, where it completes training in 18 hours 21 minutes with a peak GPU memory usage of 1.8 GB, compared to GCN, which requires 20 hours 30 minutes and 2.1 GB. Figure 4 further illustrates the convergence behavior of GCN and mGraphSAGE, showing that mGraphSAGE consistently achieves lower training and validation losses, confirming its superior learning efficiency and stability on large-scale graphs.

Table 3: Training time and peak GPU memory usage of GCN, STGCN and mGraphSAGE across different network scales

Model	12 ODs		Tiny Copenhagen		Full Copenhagen	
	Peak Memory (MB)	Training Time	Peak Memory (MB)	Training Time	Peak Memory (MB)	Training Time
GCN	122.7	6.5m	1152	25m	2066	20h30m
STGCN	1764.2	2m	10850.6	24m	-	-
mGraphSAGE	112.1	6.87m	988	23.35m	1803	18h21m



(a) Train loss

(b) Validation loss

Figure 4: Training and validation loss convergence of GCN and mGraphSAGE on the Full Copenhagen case

4.3. Prediction performance

The average prediction errors of all baseline methods, including STGCN, compared to mGraphSAGE across the 12 ODs, Tiny Copenhagen, and Full Copenhagen cases are reported in Tables 4, 5, and 6. Bold values indicate the best prediction performance. We further evaluate prediction accuracy under multiple operational scenarios that reflect different types and levels of disruption in the URT system, as described earlier.

Table 4: Results for the 12 ODs case

Methods	RR		XGB		GCN		STGCN		mGraphSAGE	
Metrics	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
All periods	3.423	2.305	3.370	2.276	3.412	2.298	**3.157**	**2.183**	3.384	2.275
Cancelations at origin >0	2.984	2.211	2.954	2.196	2.971	2.189	**2.878**	**2.119**	2.946	2.179
Cancelations at destination >0	3.197	2.368	3.166	2.358	3.195	2.355	**3.015**	**2.217**	3.171	2.352
Mean delays at origin >60s	3.254	2.321	3.184	2.282	3.219	2.304	**3.026**	**2.21**	3.202	2.289
Mean delays at origin >180s	2.629	1.946	2.539	1.87	2.565	1.907	**2.44**	**1.844**	2.598	1.931
Mean delays at origin >300s	2.115	1.661	**2.061**	**1.592**	2.081	1.623	2.213	1.759	2.094	1.646
Mean delays at destination >60s	3.058	2.174	2.982	2.125	3.013	2.146	**2.924**	**2.114**	2.991	2.131
Mean delays at destination >180s	2.358	1.716	2.341	1.681	2.331	1.680	**2.193**	**1.587**	2.317	1.671
Mean delays at destination >300s	2.296	1.682	2.306	1.697	2.292	1.680	2.319	1.693	**2.275**	**1.663**

Table 5: Results for the Tiny Copenhagen case

Methods	RR		XGB		GCN		STGCN		mGraphSAGE	
Metrics	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
All periods	0.720	0.445	0.712	0.438	0.706	0.440	0.705	0.437	**0.696**	**0.433**
Cancelations at origin >0	0.683	0.480	0.661	0.476	0.659	0.463	0.657	0.468	**0.654**	**0.457**
Cancelations at destination >0	0.672	0.468	0.659	0.465	0.654	0.452	0.654	0.462	**0.646**	**0.447**
Mean delays at origin >60s	0.675	0.441	0.635	0.417	0.631	0.416	0.634	0.418	**0.622**	**0.405**
Mean delays at origin >180s	0.618	0.446	0.566	0.401	0.572	0.411	0.568	0.400	**0.561**	**0.395**
Mean delays at origin >300s	0.662	0.485	0.590	0.421	0.586	0.422	0.585	0.412	**0.583**	**0.413**
Mean delays at destination >60s	0.724	0.476	0.705	0.456	0.707	0.464	0.698	0.454	**0.69**	**0.447**
Mean delays at destination >180s	0.632	0.449	0.607	0.421	0.594	0.411	0.599	0.413	**0.586**	**0.400**
Mean delays at destination >300s	0.667	0.501	0.625	0.453	0.611	0.444	0.619	0.444	**0.606**	**0.433**

Table 6: Results for the Full Copenhagen case

Methods	RR		XGB		GCN		mGraphSAGE	
Metrics	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
All periods	0.345	0.186	0.338	0.182	0.339	0.185	**0.332**	**0.181**
Cancelations at origin >0	0.257	0.180	0.256	0.180	0.259	0.183	**0.247**	**0.173**
Cancelations at destination >0	0.256	0.177	0.254	0.177	0.257	0.179	**0.246**	**0.170**
Mean delays at origin >60s	0.255	0.163	0.250	0.160	0.252	0.163	**0.246**	**0.158**
Mean delays at origin >180s	0.198	0.157	0.199	0.159	0.203	0.163	**0.194**	**0.154**
Mean delays at origin >300s	0.230	0.187	0.227	0.183	0.230	0.187	**0.221**	**0.178**
Mean delays at destination >60s	0.239	0.158	**0.234**	**0.155**	0.237	0.158	0.237	0.159
Mean delays at destination >180s	0.321	0.238	0.326	0.242	0.326	0.244	**0.316**	**0.236**
Mean delays at destination >300s	0.479	0.376	0.484	0.380	0.485	0.384	**0.471**	**0.370**

Overall, the results demonstrate that classical machine learning methods remain strong competitors to graph-based approaches when temporal patterns dominate the data. RR and XGB directly exploit lagged demand features and therefore capture sequential dependencies efficiently with relatively low computational overhead. However, when spatial and relational interactions among OD pairs become relevant, particularly under disruptions, explicitly modeling network structure yields clear performance advantages.

In the 12 ODs case, all methods achieve comparable performance across most scenarios due to the limited number of OD pairs and weak spatial dependencies. In this setting, demand

dynamics are largely driven by temporal regularities rather than network effects. STGCN achieves the best overall performance across both normal and disrupted periods, reflecting its strong ability to model localized spatio-temporal patterns in small-scale graphs through dedicated temporal convolution and spatial graph convolution modules. mGraphSAGE performs competitively with XGBoost and outperforms GCN and RR, which is expected given that the selected 12 OD pairs correspond to the highest-demand flows and thus exhibit strong temporal autocorrelation. Consequently, the advantages of neighborhood-based graph learning are constrained in this scenario. An additional observation is that under more severe disruptions (mean delays exceeding 300 seconds), STGCN's performance begins to degrade relative to mGraphSAGE. This suggests that while STGCN excels in small, stable networks, its fixed spatio-temporal structure may be less flexible in adapting to abrupt, system-wide propagation effects induced by large disruptions.

The benefits of graph-based learning become substantially more pronounced in the Tiny Copenhagen experiment, where spatial interactions among OD pairs are stronger and more diverse. In this setting, mGraphSAGE consistently outperforms all investigated baselines, followed by STGCN. When network effects become prominent, mGraphSAGE's ability to explicitly capture multiple relational views, including temporal similarity, spatial proximity, and origin- and destination-based interactions, provides a clear advantage. Notably, the performance gap widens as disruption severity increases: the higher the level of delay, the larger the improvement achieved by mGraphSAGE over competing methods. This can be attributed to the fact that during disruptions, OD interactions are no longer governed solely by temporal dependencies but are also shaped by network topology and operational coupling across lines and directions. By contrast, STGCN relies on a single fixed graph structure, which limits its capacity to disentangle heterogeneous propagation mechanisms. Quantitatively, mGraphSAGE improves upon STGCN by approximately 1–2.5% depending on disruption severity, while gains over RR, XGB, and GCN range from 1.5% up to 12%. Figure 5 further illustrates the improved alignment between true and predicted OD demand patterns for mGraphSAGE in this setting.

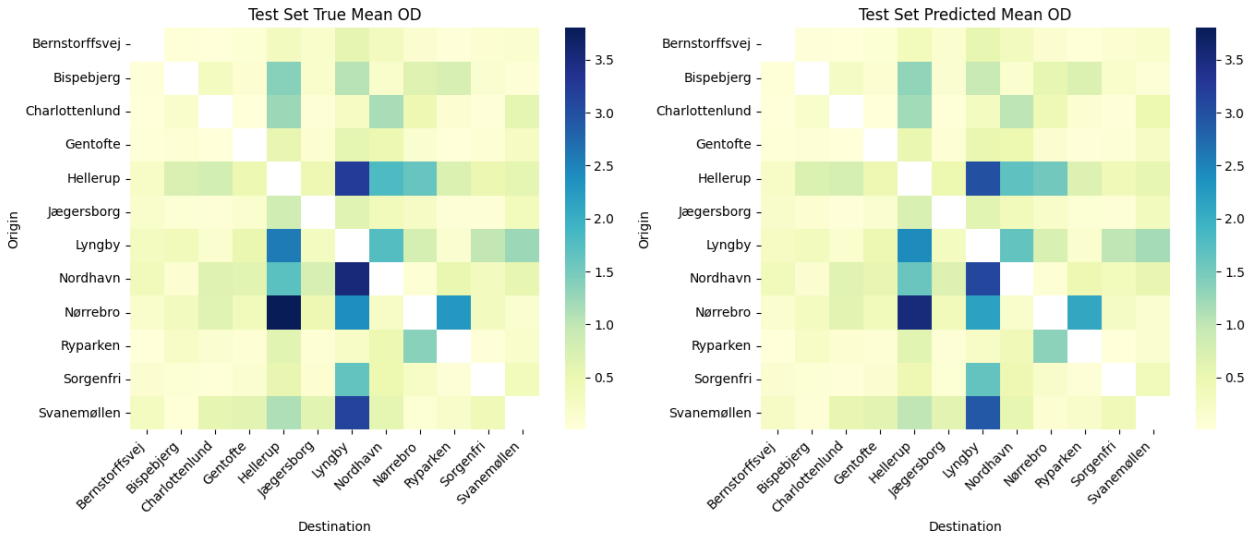


Figure 5: Average true vs. predicted OD demand in the Tiny Copenhagen test set

Across all experiments, mGraphSAGE consistently outperforms the conventional GCN baseline. Standard GCN applies uniform aggregation over full neighborhoods, which can lead to oversmoothing and limits its ability to distinguish heterogeneous relationships such as temporal similarity, spatial proximity, and directional OD flows. In contrast, mGraphSAGE decomposes these relationships into four complementary graphs and processes them through separate message-passing pathways before fusion. This design captures distinct propagation mechanisms separately, improving both accuracy and robustness. Furthermore, GraphSAGE-style neighborhood aggregation contributes to better scalability compared to full-graph convolution.

These advantages become most pronounced in the Full Copenhagen case, which comprises 6,972 OD pairs. At this scale, STGCN cannot be applied due to prohibitive memory requirements, whereas mGraphSAGE remains computationally feasible. GCN outperforms RR under normal operating conditions and achieves performance comparable to XGBoost, indicating that even basic graph aggregation can capture spatial dependencies when disruptions are limited. However, GCN's performance degrades noticeably under severe disruptions, suggesting limited robustness to abrupt changes in demand propagation. In contrast, mGraphSAGE consistently outperforms all baselines across nearly all disruption scenarios (Table 6). Despite using a simplified feature representation to manage computational complexity, mGraphSAGE achieves a 2–3% improvement over XGBoost and up to a 5% improvement over GCN during major disruptions. Only in mild disruption cases (mean delays below 60 seconds) does XGBoost marginally outperform mGraphSAGE, indicating that further enhancements in temporal modeling may yield additional gains. Figure 6 visualizes the close correspondence between the true and predicted OD demand patterns achieved by mGraphSAGE in the full-scale network. Importantly, the model not only matches the overall sparsity and magnitude distribution across OD pairs, but also reproduces the prominent vertical and horizontal bands, which reflect hub-like stations generating or attracting demand across many destinations and origins. This indicates that mGraphSAGE captures both localized OD intensities and network-level demand structure in the large-scale setting.

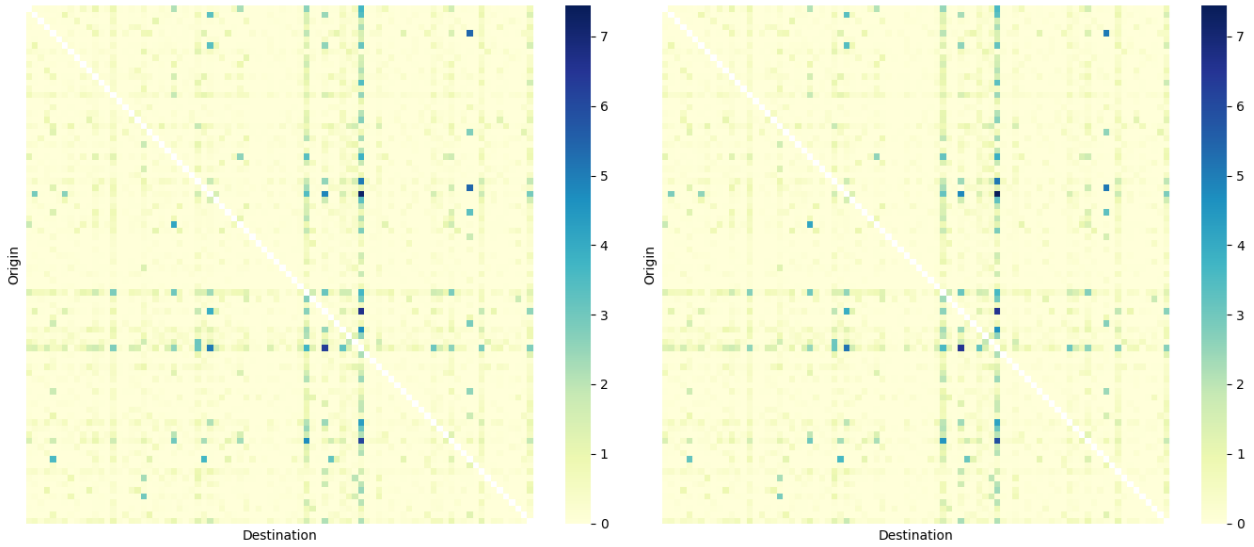


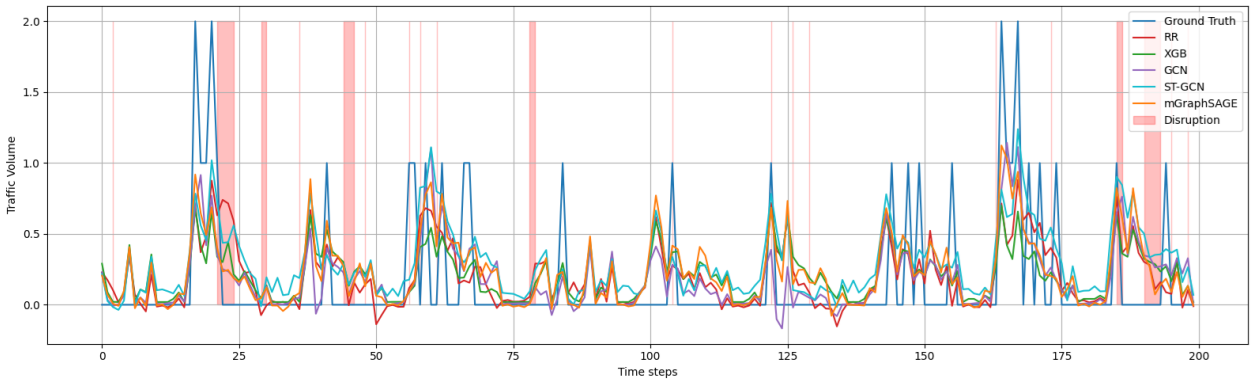
Figure 6: Average true vs. predicted OD demand in the Full Copenhagen test set.
Axes correspond to the 84 stations indexed alphabetically.

Finally, it is important to note that many OD pairs exhibit zero demand during substantial portions of the evaluation period. In this context, even modest reductions in average error correspond to meaningful improvements in prediction quality. Statistical tests confirm that the performance gains of mGraphSAGE over other baselines are significant ($p < 0.05$) in the Full Copenhagen case across all disruption scenarios, further supporting its scalability and robustness advantages.

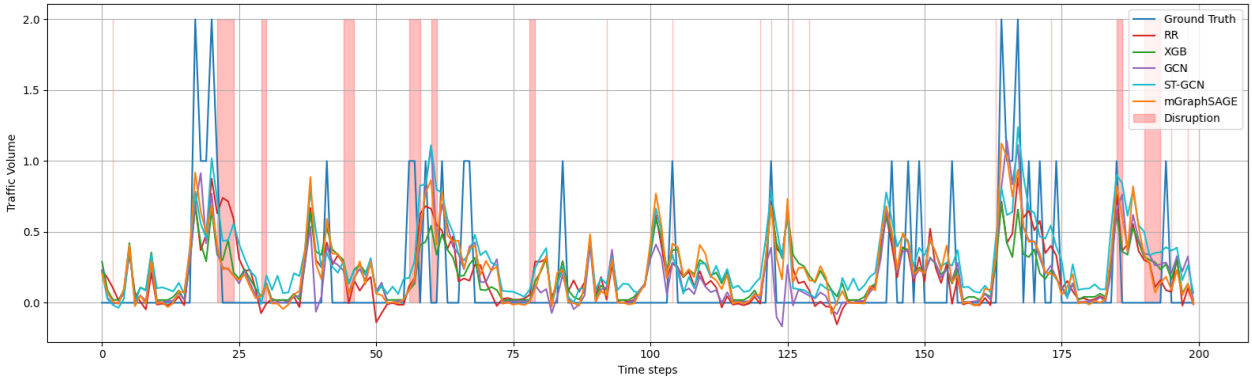
4.4. Impact of disruptions on prediction

To further examine how operational disruptions influence prediction behavior, we analyze a representative OD pair, Jægersborg–Bernstorffsvej, during periods affected by moderate but frequent disruptions. Figures 7a and 7b illustrate the predicted and actual traffic volumes over a representative 200-step horizon, focusing on periods in which the mean delay at the origin or destination station exceeds 60 seconds, highlighted by the red-shaded regions. Compared to extreme-delay cases, these conditions occur more frequently and better reflect realistic operational variability encountered in daily railway operations.

Across all baselines, prediction accuracy deteriorates during disruption periods, reflecting the increased uncertainty and nonlinearity introduced by passenger behavioral responses such as delayed departures, rerouting, or temporary demand suppression. Classical models such as RR and XGBoost tend to generate relatively smooth demand trajectories and systematically underestimate sharp peaks or rapid fluctuations. Their reliance on lagged temporal features limits their responsiveness to disruption-induced changes that are not directly encoded in recent demand history.



(a) With mean delay at the origin exceeds 60 seconds



(b) With mean delay at the destination exceeds 60 seconds

Figure 7: The predicted versus actual demand in Jægersborg–Bernstorffsvej OD with period of disruption indicated in red shaded

GCN partially mitigates this limitation by incorporating spatial aggregation, allowing it to react more quickly than purely temporal models in some disruption intervals. However, its uniform message-passing mechanism often leads to dampened responses, particularly when disruptions affect only a subset of spatially connected OD pairs. STGCN performs competitively in several intervals, benefiting from its explicit temporal modeling, but its predictions become less stable as disruptions intensify or propagate beyond localized regions.

In contrast, mGraphSAGE consistently tracks disruption-induced demand variations more closely across both origin- and destination-delay scenarios. The model exhibits improved responsiveness to sudden increases and decreases in traffic volume during disrupted periods, maintaining closer alignment with the ground truth than all baselines. This behavior can be attributed to its multi-relational graph formulation, which enables information to propagate along temporal, spatial, origin-based, and destination-based connections simultaneously. As a result, mGraphSAGE captures not only local temporal patterns but also network-wide effects arising from operational disturbances.

These results highlight two important strengths of the proposed approach. First, spatial information becomes critical even under moderate disruptions, as passenger responses propagate through interconnected OD pairs rather than remaining isolated. Second, multi-relational graph representations improve robustness, allowing mGraphSAGE to distinguish different propagation mechanisms instead of conflating them through a single adjacency struc-

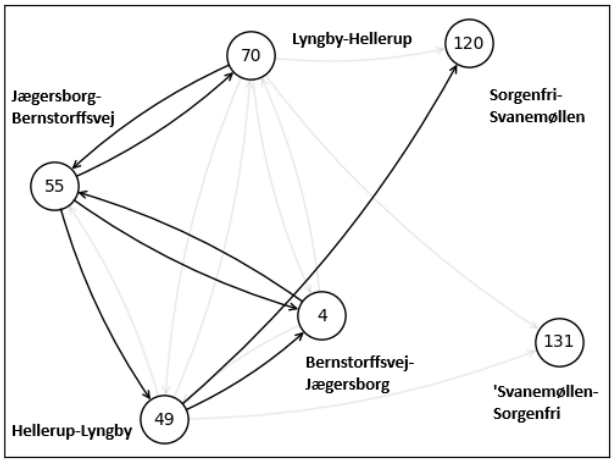


Figure 8: Spatial dependencies of Jægersborg–Bernstorffsvej OD and neigbors ODs

ture. This leads to more adaptive and realistic predictions when demand dynamics deviate from regular patterns.

Although prediction under disruption remains inherently challenging, this case study reinforces the quantitative results presented earlier. It demonstrates that mGraphSAGE provides superior robustness compared to both classical machine learning models and single-graph GNN baselines when operational uncertainty increases. These findings further confirm that enhancing graph-based spatial representations of OD relationships is a key factor in improving short-term demand prediction under real-world railway disruptions.

4.5. Interpretation for a representative OD pair

To provide deeper insight into how mGraphSAGE leverages spatial dependencies in prediction, we apply GNNExplainer [33] to the OD pair Jægersborg–Bernstorffsvej in the Tiny Copenhagen case. The explanation focuses on identifying which neighboring nodes and edges the model relies on most when predicting the demand for this OD.

Figure 8 highlights a localized subgraph centered around Jægersborg–Bernstorffsvej. The darker nodes and edges indicate higher importance scores assigned by the explainer. The strongest explanatory connections come from OD flows that are geographically and operationally related, such as Hellerup–Lyngby and Bernstorffsvej–Jægersborg. These OD pairs lie along the same line corridor and involve stations with strong commuter interactions. When compared with the actual railway topology in Figure 3a, this pattern becomes intuitive. Traveler behavior between these stations is closely linked; delays, route choices, and boarding times propagate along this stretch of the network. The model captures these relationships through the spatial graph, using directional information in edges to reflect the asymmetric influence of flows in opposite directions.

The feature importance analysis for this OD pair (Figure 9) further strengthens these insights. While temporal lag features such as previous demand and trip volumes remain influential, the top-ranked inputs include operational disruption indicators like mean delays and cancellations at the origin and destination stations. This demonstrates that mGraphSAGE does not simply rely on temporal autocorrelation; it incorporates operational dynamics that directly affect passenger behavior during disruptions. Additionally, the inclusion of

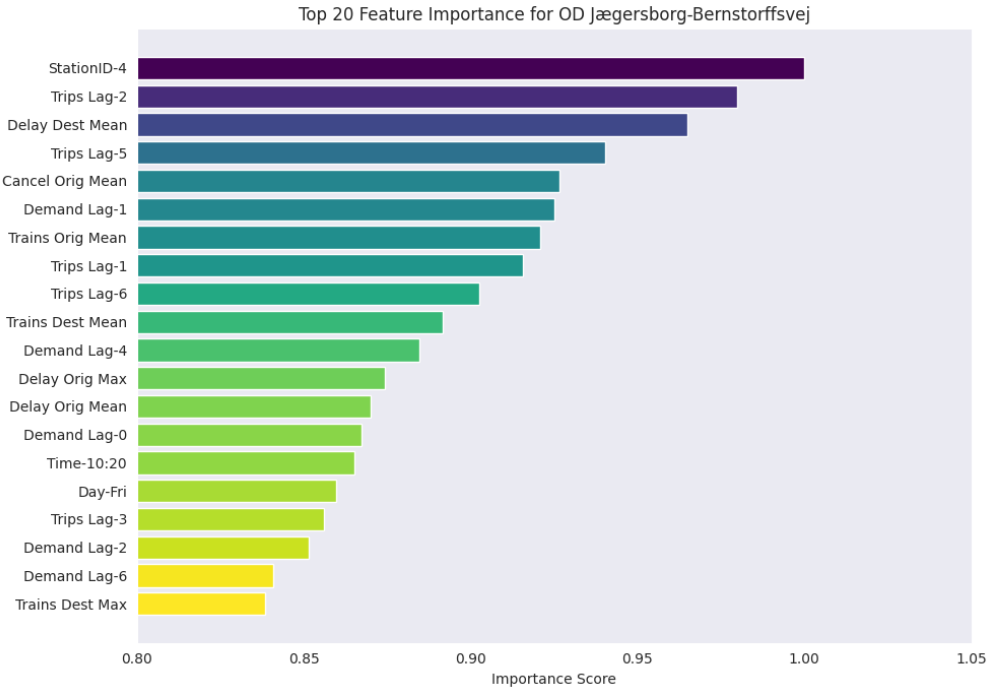


Figure 9: Feature importance analysis for Jægersborg–Bernstorffsvej OD pair

station-specific attributes (e.g., one-hot encoded origin–destination station IDs) among the most important features indicates that the model effectively captures the localized mobility characteristics and the functional roles of individual stations within the network. Calendar features such as hour-of-day and weekday indicators also contribute, reflecting recurrent travel demand cycles. Together, these findings show that mGraphSAGE integrates a rich combination of temporal, spatial, and operational inputs to infer future demand more accurately, particularly when severe delays propagate through the network.

Together, these visualizations confirm that the model learns meaningful spatial dependencies that align with the real operational structure of the rail network. Unlike classical baselines that focus solely on temporal patterns, mGraphSAGE identifies and exploits cross-OD relationships that become especially relevant under disruptions. The GNNExplainer results provide interpretability for the model's improved robustness: its predictions adjust more effectively when disruptions propagate across connected parts of the network. This demonstrates that the multi-relational graph design supports decision-making grounded in the underlying spatial and operational dynamics of urban mobility systems.

5. Conclusion

This paper introduced mGraphSAGE, a graph-based prediction model designed to enhance the spatial representation of OD demand relationships in large-scale URT networks under operational uncertainties. By representing each OD pair as a node and constructing multiple relational graph structures based on temporal similarity, spatial proximity, and origin–destination directionality, the proposed approach learns rich relational representations through inductive graph learning.

Experiments conducted on three network scales from the Copenhagen S-train system demonstrate that mGraphSAGE consistently outperforms classical machine learning methods and graph-based baselines when spatial correlations are present and become increasingly complex. While spatio-temporal models such as STGCN perform strongly in small-scale settings with localized interactions, their applicability is limited in large networks due to computational and memory constraints. In contrast, mGraphSAGE remains scalable in the Full Copenhagen scenario with 6,972 OD pairs, where conventional GCN encounters memory limitations. Training time, peak GPU memory usage, and convergence analysis further confirm the scalability and efficiency advantages of the proposed approach.

Additional evaluation under real-world disruption scenarios, including severe train delays and cancellations, shows that mGraphSAGE more effectively captures abrupt demand shifts induced by operational disturbances. By propagating information through graph connectivity, the model better reflects network-wide demand interactions during disruptions, leading to consistent performance gains over both classical and deep learning baselines. A representative case study illustrates that mGraphSAGE follows sudden demand changes more accurately than RR and XGBoost, particularly under high disruption levels.

To support interpretability and operational trust, we applied GNNExplainer to analyze predictions for a disruption-affected OD pair. The results indicate that mGraphSAGE emphasizes spatially connected OD neighbors aligned with the underlying railway topology and operational interactions. Combined with feature-importance analysis, this confirms that the model does not rely solely on historical demand lags, but also captures spatial and disruption-sensitive behavioral patterns relevant to real-world operations.

Despite these strengths, prediction performance slightly decreases in the largest-scale setting due to necessary simplifications, such as reduced node-ID encoding and constrained edge construction. Moreover, carefully tuned classical models remain competitive in scenarios dominated by temporal regularities. Nevertheless, as network-wide interdependencies grow and disruptions propagate spatially, the benefits of enhanced graph representation learning become increasingly evident.

Future work will focus on extending the proposed framework along several directions. These include: (i) evaluating mGraphSAGE on larger and more heterogeneous transit systems, (ii) incorporating additional data sources to improve demand observability, such as multimodal travel data and weather conditions, (iii) developing more realistic spatial graph formulations based on travel time, route connectivity, or passenger transfer behavior, and (iv) exploring dynamic or adaptive adjacency learning to better reflect evolving network conditions. Furthermore, integrating mGraphSAGE with explicit temporal encoders, such as recurrent networks or Transformer-based architectures, will enable a systematic assessment of how scalable graph representation learning can complement advanced temporal modeling while preserving interpretability and computational efficiency. These extensions will support the development of robust and passenger-responsive demand prediction tools for real-time operational decision-making in modern railway systems.

References

[1] Statistics Denmark, Passengers and routes - statistics denmark, `https://www.dst.dk/en/Statistik/emner/transport/persontransport/`

passagerer-og-transportruter, 2024. (Accessed on 07/24/2024).

[2] W. Jiang, Z. Ma, H. N. Koutsopoulos, Deep learning for short-term origin–destination passenger flow prediction under partial observability in urban railway systems, Neural Computing and Applications (2022) 1–18.

[3] A. Aboah, L. Johnson, S. Shah, Identifying the factors that influence urban public transit demand, 2021. URL: https://arxiv.org/abs/2111.09126. arXiv:2111.09126.

[4] J. Zhang, H. Che, F. Chen, W. Ma, Z. He, Short-term origin-destination demand prediction in urban rail transit systems: A channel-wise attentive split-convolutional neural network method, Transportation Research Part C: Emerging Technologies 124 (2021) 102928.

[5] P. Noursalehi, H. N. Koutsopoulos, J. Zhao, Dynamic origin-destination prediction in urban rail systems: A multi-resolution spatio-temporal deep learning approach, IEEE Transactions on Intelligent Transportation Systems 23 (2021) 5106–5115.

[6] S. Zhang, J. Zhang, L. Yang, F. Chen, S. Li, Z. Gao, Physics guided deep learning-based model for short-term origin-destination demand prediction in urban rail transit systems under pandemic, Engineering (2024).

[7] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, arXiv preprint arXiv:1709.04875 (2017).

[8] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, H. Li, T-gcn: A temporal graph convolutional network for traffic prediction, IEEE transactions on intelligent transportation systems 21 (2019) 3848–3858.

[9] J. Ke, X. Qin, H. Yang, Z. Zheng, Z. Zhu, J. Ye, Predicting origin-destination ride-sourcing demand with a spatio-temporal encoder-decoder residual multi-graph convolutional network, Transportation Research Part C: Emerging Technologies 122 (2021) 102858.

[10] A. Ali, Y. Zhu, M. Zakarya, Exploiting dynamic spatio-temporal graph convolutional neural networks for citywide traffic flows prediction, Neural networks 145 (2022) 233–247.

[11] W. L. Hamilton, R. Ying, J. Leskovec, Inductive representation learning on large graphs, 2018. URL: https://arxiv.org/abs/1706.02216. arXiv:1706.02216.

[12] M. Eltved, Modelling passenger behaviour in mixed scheduleand frequency-based public transport systems, Ph.D. thesis, Technical University of Denmark, 2020.

[13] A. Landex, Reliability of railway operation, in: Proceedings from the Annual Transport Conference at Aalborg University, volume 19, 2012.

[14] L. Wang, J. G. Jin, L. Sun, D.-H. Lee, Urban rail transit disruption management: Research progress and future directions, Frontiers of Engineering Management 11 (2024) 79–91.

[15] L. Liu, Y. Zhu, G. Li, Z. Wu, L. Bai, L. Lin, Online metro origin-destination prediction via heterogeneous information aggregation, IEEE Transactions on Pattern Analysis and Machine Intelligence 45 (2022) 3574–3589.

[16] G. Zhu, J. Ding, Y. Wei, Y. Yi, S. S.-D. Xu, E. Q. Wu, Two-stage od flow prediction for emergency in urban rail transit, IEEE Transactions on Intelligent Transportation Systems 25 (2023) 920–928.

[17] Z. Cheng, M. Trépanier, L. Sun, Real-time forecasting of metro origin-destination matrices with high-order weighted dynamic mode decomposition, Transportation science 56 (2022) 904–918.

[18] N. Dong, T. Li, T. Liu, R. Tu, F. Lin, H. Liu, Y. Bo, A method for short-term passenger flow prediction in urban rail transit based on deep learning, Multimedia Tools and Applications 83 (2024) 61621–61643.

[19] X. Xing, B. Wang, X. Ning, G. Wang, P. Tiwari, Short-term od flow prediction for urban rail transit control: A multi-graph spatiotemporal fusion approach, Information Fusion 118 (2025) 102950.

[20] X. Wang, D. Chen, X. Ye, X. Zhao, S. Ni, Short-term origin-destination passenger flow forecasting in urban rail transit systems: An ensemble deep learning approach based on data augmentation, Computers & Industrial Engineering (2025) 111245.

[21] Z. Duan, M. Gu, S. Luo, A multi-resolution spatiotemporal semantic learning approach for origin-destination demand prediction in urban rail transit network, Computers & Industrial Engineering (2025) 111623.

[22] A. Ali, I. Ullah, M. Shabaz, A. Sharafian, M. A. Khan, X. Bai, L. Qiu, A resource-aware multi-graph neural network for urban traffic flow prediction in multi-access edge computing systems, IEEE Transactions on Consumer Electronics (2024).

[23] A. Ali, I. Ullah, S. K. Singh, A. Sharafian, W. Jiang, H. I. Sherazi, X. Bai, Energy-efficient resource allocation for urban traffic flow prediction in edge-cloud computing, International Journal of Intelligent Systems 2025 (2025) 1863025.

[24] L. Zou, Z. Wang, R. Guo, Real-time prediction of transit origin–destination flows during underground incidents, Transportation Research Part C: Emerging Technologies 163 (2024) 104622.

[25] Q. Fan, C. Yu, J. Zuo, Predicting urban rail transit network origin–destination matrix under operational incidents with deep counterfactual inference, Applied Sciences 15 (2025) 6398.

[26] J. Zhang, Y. Zheng, D. Qi, Deep spatio-temporal residual networks for city-wide crowd flows prediction, 2017. URL: `https://arxiv.org/abs/1610.00081`. `arXiv:1610.00081`.

[27] A. Kathuria, M. Parida, C. R. Sekhar, A review of service reliability measures for public transportation systems, International Journal of Intelligent Transportation Systems Research 18 (2020) 243–255.

[28] A. Tirachini, J. Godachevich, O. Cats, J. C. Muñoz, J. Soza-Parra, Headway variability in public transport: A review of metrics, determinants, effects for quality of service and control strategies, Transport Reviews 42 (2022) 337–361.

[29] T. Akiba, S. Sano, T. Yanase, T. Ohta, M. Koyama, Optuna: A next-generation hyperparameter optimization framework, in: Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019, pp. 2623–2631.

[30] F. Rodrigues, On the importance of stationarity, strong baselines and benchmarks in transport prediction problems, in: 2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2023, pp. 4927–4932.

[31] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, CoRR abs/1603.02754 (2016). URL: http://arxiv.org/abs/1603.02754. arXiv:1603.02754.

[32] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, in: Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18, International Joint Conferences on Artificial Intelligence Organization, 2018, pp. 3634–3640. URL: https://doi.org/10.24963/ijcai.2018/505. doi:10.24963/ijcai.2018/505.

[33] Z. Ying, D. Bourgeois, J. You, M. Zitnik, J. Leskovec, Gnnexplainer: Generating explanations for graph neural networks, Advances in neural information processing systems 32 (2019).