Abstract—Accurate and interpretable delay predictions are vital for decision-making in the aviation industry. However, effectively incorporating spatiotemporal dependencies and external factors related to delay propagation remains a challenge. To address this challenge, we propose the SpatioTemporal Propagation Network (STPN), a novel space-time separable graph convolutional network that models delay propagation by considering both spatial and temporal factors. STPN uses a multi-graph convolution model that considers both geographic proximity and airline schedules from a spatial perspective, while employing a multi-head self-attention mechanism that can be learned end-to-end and explicitly accounts for various types of temporal dependencies in delay time series from a temporal perspective. Experiments on two real-world delay datasets show that STPN outperforms state-of-the-art methods for multi-step ahead arrival and departure delay prediction in large-scale airport networks. Additionally, the counterfactuals generated by STPN provide evidence of its ability to learn explainable delay propagation patterns. Comprehensive experiments also demonstrate that STPN sets a robust benchmark for general spatiotemporal forecasting. The code for STPN is available at https://github.com/Kaimaoge/STPN.

Index Terms—Delay propagation, flight delay prediction, graph neural networks, predictive models.

I. INTRODUCTION

The modern aviation industry is faced with a major challenge in the form of flight delays. In 2019, it was estimated that the global economy incurred an annual cost of approximately $50 billion as a result of flight delays, according to the International Air Transport Association [1]. However, most methods for mitigating flight delays are either too expensive or too difficult to implement. For example, building new airports to ease congestion is a daunting task, with the average cost for a commercial airport being $30 million per 3 km runway and $500 per square meter for a passenger terminal [2]. With the rise of information technology and access to vast flight data, the field of flight delay prediction has emerged. Advanced air traffic management techniques such as short-term and long-term traffic management techniques such as short-term and long-term

Fig. 1. Spatiotemporal dependencies within the airport network.
1) Exogenous Factors: The nature of air transportation operations makes it susceptible to external factors, such as extreme weather like high winds, low visibility, and thunderstorms. These conditions are the main source of flight delays, causing nearly 40% of flight disruptions [7].

2) Multi-relational Spatial Dependencies: The flight delays in the air transportation network often exhibit both local similarities between neighboring airports, due to the local impact of external factors, and long-range spatial dependencies, as delayed flights can spread the effects of delay to airports far apart from each other.

3) Coupled Spatiotemporal Effects: The propagation of delays in an airport network is facilitated by flights connecting different airports. The resulting delay between two airports is influenced by both the travel time of the flight and the distance between the airports, leading to a coupling of spatial and temporal dependencies in the delay propagation process [8].

4) Departure-Arrival Delay Relationship: A single delayed flight can result in both departure and arrival delays, and if the delay is significant, it can impact related flights at multiple airports. This highlights the interconnected nature of the airport network, where arrival and departure delays can simultaneously propagate and amplify through the network.

Given the aforementioned complexity of airport network delay, delay prediction has been a hot topic for decades, falling into two main categories: knowledge-driven and data-driven approaches. In transportation and operational research, previous works have focused on modeling the dynamics of delay propagation using queuing theory [9], [10]. The knowledge-driven approaches are explainable and help identify factors that mitigate or amplify delay propagation. However, flight delay data is routinely produced in high-volume and high-dimension and can not be easily handled via classical knowledge-driven approaches. Several works have applied machine learning algorithms including random forest, Gradient Boosting Regression Tree (GBRT), and K-nearest neighbor algorithm [11] for the single airport and network-wide delay prediction. A problem with traditional machine learning methods is that their shallow structure can not efficiently handle spatial dependencies within Big Data. Most recently, deep learning models for delay prediction have been developed [12], [13], [14]. However, those approaches only considered limited factors using a simple structure without incorporating the aforementioned characteristics including the multi-relational spatial dependencies, coupled ST effects, and departure-arrival delay relationships.

This paper investigates the concurrent prediction of multi-step arrival and departure delay in a large airport network. To account for the intricate ST relationships within the airport network, we utilize a multi-relational graph representation, where the nodes correspond to airports, and the edges are weighted based on flight volume and airport distances. The delay propagation process within this graph is modeled using a Space-Time-Separable Graph Convolutional Network (STSGCN) [15]. To further improve the model, we introduce the SpatioTemporal Propagation Network (STPN), which integrates the self-attention mechanism [16] for temporal dependency learning, diffusion convolution [17] for spatial dependency capture, and squeeze-and-excitation [18] for feature relationship modeling. In real-world delay datasets, STPN outperforms existing traffic forecasting models, demonstrating its superiority. Moreover, it also provides a competitive baseline for general ST forecasting tasks. To summarize, this paper presents a novel approach for predicting multi-step arrival and departure delay in a large airport network, which outperforms existing state-of-the-art baselines. In summary:

- Our model is based on a space-time separable graph convolution network approach. Unlike traditional models, we utilize multiple graph structures in both time and space modes. Theoretical analysis through tensor algebra demonstrates that our model learns a sum of Kronecker products kernels, instead of a single Kronecker kernel [15]. This enables our model to capture more intricate ST dependencies.
- Our approach models the complex spatial dependencies through a random walk process on a multi-relational graph. This process is characterized using diffusion graph convolution. We incorporate a multi-head self-attention mechanism to model the temporal dependencies in our model. The mechanism allows us to learn multiple temporal adjacency matrices, enabling us to capture diverse temporal dependencies simultaneously.
- Our proposed method, the SpatioTemporal Propagation Network (STPN), has been thoroughly tested on two large-scale datasets and has demonstrated its effectiveness in predicting arrival and departure delays. Additionally, the generation of counterfactuals by STPN provides evidence of its ability to uncover and understand the patterns behind delay propagation. Furthermore, while STPN was specifically designed for forecasting airport delays, it has achieved comparable results in other ST traffic forecasting tasks.

The rest of this paper is organized as follows. First, we review some related works of flight delay propagation modeling and graph neural networks for ST prediction in Section II. We then introduce the proposed approach for network-level flight delay prediction in Section III. Extensive comparison, ablation analysis, and counterfactual intervention are conducted in Section IV. Extension of STPN to other ST forecasting tasks are shown in Section V. Finally, we conclude this paper in Section VI.

II. RELATED WORKS

A. Network-Wide Delay Propagation Modeling

The earliest study on flight delay propagation can date back to 1998 [19]. The propagated delay occurs because of connected resources involved in an initially delayed flight and flights downstream. Using the flight schedule, Beatty et al. [19] construct a delay tree containing 50–75 connecting flights. Later, this simple approach was extended to study network-wide airport congestion [20]. The most successful analytical model for studying delay propagation is the Approximate Network Delays model (AND) [9], which employs a combination of a queuing model for simulating initial delays and a delay-propagation algorithm. The AND model mainly uses flight schedules to simulate delay propagation and can uncover rich temporal dependencies of
airport delays. Then, several applications [21], and improvements [6], [22] under the umbrella of AND are proposed. Other analytical models like multivariate simultaneous equation regression show that major airports have a higher impact on delay propagation [23].

A recent study [7] has suggested that graph signal processing is a promising tool for studying delay propagation. For instance, airport delays can be treated as node signals in a graph, and various ST patterns can be analyzed by graph spectral analysis. Some researchers have also utilized GCNs to learn delay propagation. For example, Bao et al. [13] proposed AG2S-Net, which models spatial dependencies as learning parameters, while Cai et al. [14] proposed MSTAGCN, which uses feature embeddings to weight the spatial interactions between different airports. However, a drawback of these models is that they treat spatial and temporal dependencies separately, even though the two are closely related.

### B. Graph Neural Networks for Spatiotemporal Prediction

GNNs are divided into two main categories, the spectral-based approaches and spatial-based approaches [24]. Most existing ST prediction frameworks are based on the spectral-based Graph Convolutional Network (GCN), which is initially proposed by Bruna et al. [25]. Since then, several ideas [26], [27], [28], [29], [30] have been proposed to improve the performance of GCN.

In the context of ST prediction, the GNN-based approaches treat sensors or locations as nodes of a graph, and establish edges according to their spatial relationships. Then the spectral-based GCNs are utilized to capture the spatial dependencies of the established graph signals. To simultaneously model the temporal dependencies, GCNs are combined with recurrent neural networks (RNNs), temporal convolutional networks (TCNs), and self-attentional mechanism. Seo et al. [30] used GCNs to filter inputs and hidden states in RNNs. Later, Li et al. [17] combined RNNs with diffusion convolution for long-term traffic forecasting, in which the effects of asymmetric spatial dependencies can be taken into account. To reduce the computational cost brought by RNNs, Yu et al. [31] built a complete convolutional structure named STGCN, in which a specifically designed TCN is used to capture the temporal dependencies. The earlier ST prediction approaches were limited to graphs constructed by geographic proximity and incapable of uncovering other types of spatial dependencies. Wu et al. [32] mitigated this issue using a self-adaptive adjacency matrix. As a result, this enabled capturing spatial dependencies from farther sensors, but it did not consider dynamic spatial dependencies. To address this drawback, Zhang et al. [33] introduced the GMAN model that dynamically assigns different attention weights to different sensors at different time steps. Guo et al. [34] further combined dynamic graph convolution with Transformer to capture the temporal dynamics of ST data. As opposed to previous efforts, Sofianos et al. [15] used a pure GCN structure STSGCN to handle spatial and temporal dependencies. They factorized ST graphs into space and time adjacency matrices, which are end-to-end learnable. Their method achieves state-of-the-art performance on human pose forecasting and gives us a lot of inspiration. While our model has gained much understanding from STSGCN, it is fundamentally different. STSGCN solely learns its spatial and temporal adjacency matrices, whereas in our model, the spatial adjacency matrix is defined using prior knowledge and the temporal adjacency matrix is learned through the self-attention mechanism. Another related work is the attempt [35] to generalize multiple graph convolution on matrix completion tasks, in which two GCNs are applied to each dimension of the matrix.

### III. METHODOLOGY

The proposed model uses historical departure/arrival delay data and external factors such as weather to predict future departure/arrival delays in the long term. The STSGCN is used to learn the ST propagation patterns between airports, taking into account both space and time interactions. The framework of STPN is demonstrated in Fig. 2. The proposed STPN architecture utilizes multiple Space-Time-Separable Multi-Graph Convolution layers and Squeeze-and-Excitation layers, stacked on top of each other. The model takes into account the influence of geographic proximity, weather, and traffic volume on delay propagation. This section delves deeper into the working mechanism of the STPN model.

#### A. Problem Formalization

The airport network delay prediction problem in this paper is the following. Given a set of $N$ airports, we represent those airports as a weighted multi-relational graph $G = \{V, E, R\}$, where $V$ is a set of nodes $|V| = N$, $E$ is a set of $Q$ relations, and $R \subseteq V \times V \times R$ is a set of $m$ weighted edges. Given $Q$ relations, we can define $Q$ adjacency matrices $A = [A_1, \ldots , A_Q]$ with $A_q \in \mathbb{R}^{N \times N}$. We denote the arrival and departure delays by 2-dimensional vectors $x_{v,k}$ representing delays of airport $v$ at time $k$, the covariate (etc. air condition) vectors $z_{v,k}$ representing weather type of airport $v$ at time $k$. Denote the delays observed on graph $G$ as a matrix $X \in \mathbb{R}^{N \times 2}$, and the covariates as a matrix $Z \in \mathbb{R}^{N \times C}$ with $C$ categories. The delay historical observation from time point $t - h + 1$ to $t$ is denoted by a $3$-D tensor $X^{t-h+1:t-1} = [X^{t-h+1}, \ldots , X^t] \in \mathbb{R}^{N \times h \times 2}$, and historical covariates are denoted by a $3$-D tensor $Z^{t-h+1:t-1} = [Z^{t-h+1}, \ldots , Z^t] \in \mathbb{R}^{N \times h \times 2}$. The aim is to learn a function $f(\cdot)$ that maps $h$ historical observations and covariates to future $p$ delays, given a multi-relational graph $G$(1).

$$f(\cdot) : (X^{t-h+1:t-1}, Z^{t-h+1:t-1}, G) \rightarrow [X^t, \ldots , X^{t+p}] \quad (1)$$

In the following section, we will show how we use a space-time separable graph neural networks to model the mapping $f(\cdot)$. To make our model more comprehensible, the important notation is given in Table I.

#### B. Space-Time Separable Graph Convolution

The STPN model is partial inspired by STSGCN [15]. To make the proposed STPN more understandable, we use tensor algebra to emphasize STSGCN. The input of the traditional graph convolution layer $l$ is a matrix $H^{(l)} \in \mathbb{R}^{N \times C^{(l)}}$, where $N$ is the number of nodes, and $C^{(l)}$ is the number of features. The graph convolution layer $l$ outputs the $H^{(l+1)} \in \mathbb{R}^{N \times C^{(l+1)}}$.  

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
Fig. 2. The STPN architecture utilizes multiple Space-Time-Separable Multi-Graph Convolution layers and Squeeze-and-Excitation layers, stacked on top of each other. The output layer is a Space-Time-Separable Multi-Graph Convolution layer that incorporates a future time point query vector. STPN processes a sequence of past departure/arrival delay and weather data for an airport network. It employs self-attention to identify temporal dependencies and leverages geographic proximity and historical origin-destination traffic volume to enable diverse spatial propagation.

TABLE I

| Notation | Description |
|----------|-------------|
| \( \mathcal{G} \) | The graph structure |
| \( X \) | The network-wide arrival-departure delay |
| \( Z \) | Covariate variable associated with delay |
| \( H \) | Hidden state of neural networks |
| \( A \) | Adjacency matrix |
| \( \mathcal{H} \) | Tensor view of the hidden state |
| \( W \) | Learable weights |

given by the following

\[
H^{(l+1)} = \sigma \left( AH^{(l)}W^{(l)} \right),
\]

where \( A \in \mathbb{R}^{N \times N} \) is the normalized adjacency matrix, \( W^{(l)} \in \mathbb{R}^{C^{(l)} \times C^{(l+1)}} \) are the trainable weights of layer \( l \), and \( \sigma \) is the activation function.

Different from the basic graph convolution layer, the input of the STSGCN layer is a tensor \( H^{(l)} \in \mathbb{R}^{N \times T^{(l)} \times C^{(l)}} \). We define STSGCN via tensor mode product [36]:

**Definition 1:** The \( n \)-mode (matrix) product of a tensor \( \mathcal{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n} \) with a matrix \( U \in \mathbb{R}^{I_n \times J} \) is denoted by \( \mathcal{X} \times_n U \) is of size \( I_1 \times \cdots \times I_{n-1} \times J \times \cdots \times I_N \). Elementwise, we have

\[
(\mathcal{X} \times_n U)_{i_1, \ldots, i_{n-1}, j, i_{n+1}, \ldots, i_N} = \sum_{i_n=1}^{I_n} x_{i_1, \ldots, i_{n-1}, i_n, i_{n+1}, \ldots, i_N} u_{j, i_n}.
\]

For input tensor \( H^{(l)} \), we have space (S), time (T), and feature (F) modes. The graph convolution of STSGCN can be defined as follows:

\[
H^{(l+1)} = \sigma \left( \mathcal{H}^{(l)} \times_S A^{(l)}_S \times_T A^{(l)}_T \times_F W^{(l)} \right),
\]

(3)

where \( A^{(l)}_S \in \mathbb{R}^{N \times N} \) and \( A^{(l)}_T \in \mathbb{R}^{T^{(l)} \times T^{(l+1)}} \) are spatial and temporal adjacency matrices, respectively, and \( H^{(l+1)} \in \mathbb{R}^{N \times T^{(l+1)} \times C^{(l+1)}} \) is the output. The tensor form given in (3) is equivalent to the following matrix form

\[
H^{(l+1)} = \sigma \left( \left( A^{(l)}_S \otimes A^{(l)}_T \right)^T H^{(l)} W^{(l)} \right),
\]

(4)

where \( H^{(l)} \in \mathbb{R}^{NT^{(l)} \times C^{(l)}} \), \( \otimes \) is the Kronecker product combining \( A^{(l)}_S \) and \( A^{(l)}_T \) into an \( NT^{(l)} \times NT^{(l+1)} \) matrix block. \( A^{(l)}_S \otimes A^{(l)}_T \) is the Kronecker product kernel, widely used in Gaussian Processes (GPs) [37]. It reduces the model parameters and avoids inference of full ST kernel. However, the Kronecker product kernel assumes the spatial dependencies at best change by a factor in time, hence is unable to characterize more complex ST dependencies [38].
The original STSGCN [15] treated $A_T^{(l)}$ and $A_S^{(l)}$ as purely trainable weights for neural networks. However, this inevitably adds learning parameters to the neural networks, which is not feasible for graphs with a large number of nodes. Moreover, learning $A_S^{(l)}$ and $A_S^{(l)}$ with fixed sizes makes the neural networks transductive, and full retraining is required when a new node is added to the graph (such as a newly constructed airport in the system) [39]. Another drawback is that the purely learned adjacency matrix neglects prior knowledge and external factors related to ST dependencies. In order to enhance the generalization capability of STPN, we use an inductive architecture to infer ST dependencies, making full use of prior knowledge about the aviation system. The following sections will introduce the methods we use to learn ST dependencies in the proposed STPN.

C. Self-Attention for Temporal Mode

The self-attention layer for temporal mode is to obtain a representative temporal adjacency matrix $A_T^{(l)}$. To account for the daily periodicity and rhythms of flight schedule, we introduce positional encoding [16] for the time of day $t$

$$pe_{pos,n,2i} = \sin \left( \frac{pos_n L_{pos}^{2i}}{\text{d}} \right),$$

$$pe_{pos,n,2i+1} = \cos \left( \frac{pos_n L_{pos}^{2i}}{\text{d}} \right),$$

where $pos_n \in \{0,1,\ldots,t_d\}$ is the time of day, $t_d$ is the maximum value of daily time determined by the temporal resolution of delay data, $d$ is the total dimension of the embedding, $L_{pos}$ is the scaling factor. Although a fixed $L_{pos}$ (e.g., 10,000) is often selected in defining positional encoding, we treat it as a learning parameter because it could be beneficial for general approximation [40].

Given a series of historical delays $X$ and their associated embedding $P = \{pe_{t-h+1}, \ldots, pe_{t+1}\} \in \mathbb{R}^{h \times J}$. We compute the temporal adjacency matrix $A_T^{(l)}$ by employing the self-attention mechanism. The multi-head self-attention layer transforms the embedding $P$ into query matrix $Q^{(l)} = PW^{(l)}_Q$ and key matrices $K^{(l)} = PW^{(l)}_K$, where, $W^{(l)}_Q, W^{(l)}_K \in \mathbb{R}^{J \times c_k}$ are learnable parameters of the l-th layer. After these linear projections, the scaled dot-product attention computes the adjacency matrix

$$A_T^{(l)} = \text{softmax} \left( \frac{Q^{(l)}(K^{(l)})^\top}{\sqrt{c_k}} \right).$$

We apply self-attention to the hidden layer of STPN. Since the attention mechanism is order-independent, we can provide positions for any time of day and compute the associated adjacency matrix. For the output layer, the target is to obtain $[X^1, \ldots, X^{h+p}]$, and we have positional embedding $P^* = \{pe_1, \ldots, pe_{h+p}\} \in \mathbb{R}^{p \times J}$. The query for the output layer will be $Q^o = P^*W^{(l)}_Q$, then we have $A_T^{(l)} \in \mathbb{R}^{h \times p}$. Using tensor mode product $H_T \times_T A_T^{(l)}$, the temporal size of the outputs will change from $h$ to $p$. We can also generate multiple temporal adjacency matrices using (6) to capture multiple temporal dependencies better, it will yield multiple temporal representations $[H \times_T A_T^{(1)}, \ldots, H \times_T A_T^{(l)}]$ (multi-head attention). We will use these representations jointly with multiple graphs of spatial mode for modeling ST dependencies.

D. Multi-Graph Convolution for Spatial Mode

We consider that the spatial dependencies of flight delay arise from a multi-relational graph and relate the spatial propagation of delay to random walks. Let the vector $p^t \in \mathbb{R}^N$ denote the delay probability distribution on a multi-relational graph, $p^t_n$ indicate the probability of being at node $n$ at time $t$. The probability of taking random walk according to relation 1, $w_q$ denote the probability of taking random walk. To derive $p^{t+1}$ from $p^t$, the random walk process can be stated as

$$p^{t+1} = \sum_{q=1}^{Q} w_q A_q p^t,$$

where $A_q$ represents the power series of the transition matrix with $A_q = A_q / \text{rowsum}(A_q)$. This process can be modelled by a diffusion convolution layer, which proves to be effective in spatial-temporal modeling [17], [32]. The diffusion convolution layer can be generalized to the following equation

$$X *q = \sum_{k}^{K} \sum_{q}^{Q} \tilde{A}_k^q X W_{q,k},$$

where $K$ is the number of diffusion steps, $W_{q,k}$ are learnable weights. Using multiple temporal representations jointly with diffusion convolution, we yield the following tensor algebra form

$$H_{(l+1)} = \sigma \left( \sum_{k}^{K} \sum_{q}^{Q} \sum_{i}^{I} H^i_T \times_T \tilde{A}_k^q \times_T A_T^{(l)}_i \times_F W_{q,k} \right).$$

Ignoring the learnable weights $W^{(l)}_q$ on feature mode, we have the following matrix form

$$\sum_{k}^{K} \sum_{q}^{Q} \sum_{i}^{I} \left( \tilde{A}_k^q \otimes A_T^{(l)}_i \right)^\top H^i_T,$$

(10)

$$\sum_{k}^{K} \sum_{q}^{Q} \sum_{i}^{I} \left( \tilde{A}_k^q \otimes A_T^{(l)}_i \right)^\top = \text{equal to the sum of}$$

Kronecker products kernel in Gaussian Processes [41]. In the Sum of Kronecker (SoK) products kernel, each term $\tilde{A}_k^q \otimes A_T^{(l)}_i$ presents a combination of one spatial and temporal dependencies. Noted that the STSGCN [15] uses a single Kronecker product kernel to model ST correlations in the data. However, this approach is limited in capturing complex interactions between spatial and temporal dimensions in datasets like airport delay, where the spatial and temporal dimensions are not independent. The SoK kernel used in STPN overcomes this limitation by allowing for multiple temporal evolutions with specific spatial patterns. The expression $\sum_{k}^{K} \sum_{q}^{Q} \sum_{i}^{I} \left( \tilde{A}_k^q \otimes A_T^{(l)}_i \right)^\top$ in (10) can be interpreted as the coefficients of a regression equation that denotes the contribution of each ST point to the output. By modeling these coefficients
using the Sum-of-Kronecker kernel, we can effectively capture the ST correlation between features. For instance, the correlation between two points \( x = (x_v, x_t) \) and \( y = (y_v, y_t) \) is defined as:

\[
C(x, y) = \sum_k \sum_q \sum_l \left( \hat{A}_k(x_v, y_v) \cdot A_q^l(x_t, y_t) \right),
\]

where the Kronecker product kernels are summed over all possible pairs of features. This approach allows the resulted kernel to capture complex interactions between spatial and temporal dimensions that cannot be captured by a simple dot product or other kernel functions that consider each dimension independently.

E. Squeeze-and-Excitation on Feature Mode

In our STPN, the arrival delays, departure delays, and embeddings of the weather category are directly treated as features of the graph neural networks. Those features are related to each other. In the hidden layer \( l \), we have a feature map tensor \( H(l) \) of size \( N \times T(l) \times C(l) \) with \( C(l) \) feature maps. The relationship between different features are captured by fully connected layer with learnable weights \( W_q^{l,k} \). We assume that these feature maps are redundant and have a different magnitude of importance for delay prediction. To make STPN more sensitive to informative features, we add a squeeze-and-excitation (SE) block [18] on feature mode. Given the feature map \( H(l) \), we have squeeze vector \( z_c^{(l)} \in R^{C(l)} \) whose \( c \)-th value equals to:

\[
z_c^{(l)} = \frac{1}{N \times T(l)} \sum_{v=1}^{V} \sum_{j=1}^{T(l)} H_{c,v,k}^{l}. \tag{11}
\]

Then we have a gating mechanism:

\[
s_c^{(l)} = \text{Sigmoid} \left( W_{SE1}^{l} \cdot \text{ReLU} \left( W_{SE2}^{l} z_c^{(l)} \right) \right), \tag{12}
\]

where \( W_{SE1}^{l} \in \mathbb{R}^{C(l) \times C(l)} \) and \( W_{SE2}^{l} \in \mathbb{R}^{C(l) \times C(l)} \). Finally, the \( c \)-th feature map is produced by:

\[
H_c^{l} = s_c^{(l)} \cdot H_c^{l}. \tag{13}
\]

In our experiments, we find that the SE block can slightly improve our model’s performance. To this end, our STPN consists of several space-time-separable multi-graph convolution layers given in (9) with residual connections PReLU activation followed by an SE block.

F. Training

The proposed architecture is trained end-to-end supervisely. The model is trained by Root Mean Squared Error (RMSE)

\[
\mathcal{L}_{RMSE} = \sqrt{\frac{1}{Npc} \sum_{n=1}^{N} \sum_{i=0}^{p} \sum_{c=1}^{2} (x_{n,t+i,c} - \hat{x}_{n,t+i,c})^2}. \tag{14}
\]

It should be noted that some airports do not have flights for some periods. We treat those data points as missing data and mask their training losses.

IV. EXPERIMENTS

In this section, we assess the quality of delay prediction in two different datasets and perform ablation studies to isolate the impact of each building block in the proposed regime.

A. Evaluation Metrics

Throughout all experiments, we predict both network-wide arrival and departure delay. This yields delay observation \( x_{v,k} \in \mathbb{R}^2 \) at time point \( k \) on airport \( v \). We calculate the errors for arrival and departure delays separately.

To evaluate the quality of delay prediction, we use three common metrics: mean absolute error \( MAE \), root mean squared error \( RMSE \), R squared \( R^2 \). Letting \( J \) denote the prediction set, \( \hat{x} \) the prediction, \( x \) the true observations, the three scores are defined as:

\[
RMSE(x, \hat{x}) = \sqrt{\frac{1}{|J|} \sum_{j \in J} (x_j - \hat{x}_j)^2},
\]

\[
MAE(x, \hat{x}) = \frac{1}{|J|} \sum_{j \in J} |x_j - \hat{x}_j|,
\]

\[
R^2(x, \hat{x}) = 1 - \frac{\sum_{j \in J} (x_j - \hat{x}_j)^2}{\sum_{j \in J} (x_j - \text{mean}(x))^2}. \tag{15}
\]

B. Datasets

We use two publicly available delay data to experiment with network-wide airport delays of different characteristics. The first U.S. delay dataset is collected from the U.S. Bureau of Transportation Statistics (BTS) database (https://www.transtats.bts.gov/DL SelectFields.asp?gnoyr VQ=). Seven-year flight data from January 1st, 2015, to December 31st, 2021, are collected. The initially collected dataset includes 360 airports. We select 70 airports with heavier traffic volumes for our experiments. The U.S. weather dataset of the same airports during the same time period is obtained from [42]. Eight weather categories, which include normal weather, severe cold, fog, hail, rain, snow, storm, and other precipitation, are considered. The second China delay dataset is collected from Xiecheng (https://pan.baidu.com/s/1dEPyMGh#list/path=%2F). Two-year flight data from April 30th, 2015, to May 1st, 2017, are collected. Seven weather categories, which include normal weather, rain, cloud, thunderstorm, fog, storm and snow, are obtained from the associated special event data. Only flight records between 6 am to 12 pm are considered for those two datasets because very few flights are observed outside this period.

We aggregate the flight’s arrival and departure delays into 30 minutes windows according to their origin and destination in both datasets. Consequently, the exact value of flights with delays higher than 30 minutes can not be obtained in real-time. We use the following equation to compute the average arrival and departure delay \( (a_{v,k}, d_{v,k}) \) of airport \( v \) at time point \( k \):

\[
a_{v,k} = \frac{\sum_{i \in V} \sum_{j \in K} \min(d_{v,i,j}^0, 30)}{|V||K|},
\]

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.
\[ d_{i,k} = \frac{\sum_{j\in V'} \sum_{k'=1}^{d_{i,j}} \min(d_{i,j}', 30)}{|V'||K'|}, \quad (16) \]

where \( V \) and \( V' \) are the set of flights whose destination and origin are at airport \( v \), respectively, \( K \) and \( K' \) are the set of flights whose schedule arrival and departure time are during \([k, k+30)\), \( d_{i,j}' \) and \( d_{i,j}'' \) is the corresponding flights’ arrival and departure delays. The form in (16) directly illustrates the delay level, with a maximum value of 30 min. After aggregation, the U.S. delay dataset contains 16% unobserved data (missing data or data point without a flight), the China delay dataset contains 42% unobserved data. It should be noted that in our analysis, we excluded all cancelled flights and only included flights with both exact arrival and departure times.

To construct the multi-relational graph of airports, we use three types of adjacency matrices. Like typical ST prediction models [17], [31], we compute the pairwise distances between airports and build the distance adjacency matrix \( A_d \) using threshold-olded Gaussian kernel \( A_d(i,j) = \exp(-\frac{\text{dist}(i,j)^2}{\sigma^2}) \), and we set \( A_d(i,j) = 0 \) if \( A_d(i,j) \leq 0.1 \). In addition, we also consider the origin-destination (O-D) and destination-origin (D-O) relations. The O-D adjacency matrix is computed by

\[ A_{O\rightarrow D}(i,j) = \begin{cases} 0, & \text{if } F_{i\rightarrow j} < 0.15 \hat{F}_{O\rightarrow D}, \\ \frac{1.5}{F_{i\rightarrow j}} \hat{F}_{O\rightarrow D}, & \text{otherwise}. \end{cases} \quad (17) \]

where \( F_{i\rightarrow j} \) is the total air flow from airport \( i \) to \( j \) of the training dataset, \( \hat{F}_{O\rightarrow D} \) is the maximum value of OD flow pair in the training dataset. The destination-origin adjacency matrix \( A_{D\rightarrow O} \) can be directly computed by \( A_{O\rightarrow D} \) transposed.

C. Experimental Setups

We conduct multi-step ahead delay prediction for both two datasets. For the U.S. delay dataset, we use 12 previous time points (6 hours) to predict delays of 12 future time points (6 hours). For the China dataset, we use 36 previous time points (18 hours) to predict delays of 12 future time points (6 hours) because it contains more missing data. Z-score normalization is applied to inputs. The datasets are split in chronological order with 70% for training, 10% for validation and 20% for testing.

We compare STPN with widely used ST prediction models, including (1) HA: Historical Average, which models the airport delay as a seasonal process, and uses the weighted average of previous seasons as the prediction. The period used is one week, and the prediction is based on aggregated data from previous weeks. (2) VAR: Vector Auto-Regression. The lags are set to 12 for the U.S. delay dataset and 36 for the China delay dataset. (3) Long Short-term Memory (LSTM): The Encoder-decoder framework using LSTM with a peephole. Both the encoder and the decoder contain two recurrent layers. In each recurrent layer, there are 256 units. The model is trained with batch size 64 and loss function MAE, the learning rate is 0.01, and the early stop is performed by monitoring the validation error. The LSTM model does not utilize spatial correlation. We train one LSTM model to predict departure and arrival delays for all airports. (4) SpatioTemporal Graph Convolutional Network (STGCN) [31]: The graph convolution layer of STGCN only contains one graph.

We use the O-D adjacency matrix given in (17) to perform graph convolution. The channels of three layers in ST-Conv block of STGCN are 64, 16, and 64, respectively. The graph convolution kernel size and temporal convolution kernel size are set to 3 in the model. We train STGCN by minimizing the root mean square error using Adam for 50 epochs with a batch size of 32. The learning rate is 0.001. (5) Graph WaveNet (Gwave) [32]. We use eight layers of Graph WaveNet. We use (8) as our graph convolution layer with a diffusion step 2. We train Gwave using Adam optimizer with an initial learning rate of 0.001. The STGCN and Gwave models are originally used for single variable forecasting. We replace their output layer with a fully-connected layer that outputs two variables. (6) Space-Time-Separable Graph Convolutional Network (STSGCN) [15].\(^1\) We use three STSGCN layers, and the kernel size for temporal convolution is set to [3,3,3].

We implement the STPN model based on the PyTorch\(^2\) framework. We choose Root Mean Square error as the loss function. The hyperparameters and the best models are determined by the performance of the validation sets. We use four layers of space-time-separable multi-graph convolution in (9), and the last layer is the output layer. The feature dimension of the hidden layer is [128, 64, 32]. Each hidden layer is followed by a SE block with the reduction rate \( r = 16 \). The number of the head for the attention model is 4. The order of graph convolution is 2. We use a 4-dimensional embedding to encode weather data. We train STPN using Adam optimizer with a learning rate of 0.001.

D. Main Results and Analysis

The results for the U.S. and China delay datasets are summarized in Tables II and III. Based on the results, several observations can be made. First, graph neural networks, including STGCN, STSGCN, Gwave, and STPN, generally outperform other baselines on the U.S. delay dataset, validating the premise of this work that the graph structure can be used to capture the spatiotemporal propagation patterns of airport delays. Second, among the graph-based methods, STPN outperforms STGCN, STSGCN, and Gwave. STPN utilizes the self-attention mechanism instead of temporal convolutional networks to capture temporal dependencies and employs the SE block to model feature relationships, enabling it to better capture ST dependencies and arrival-departure interactions within the delay datasets. In contrast, STSGCN uses purely learnable adjacency matrices to learn ST dependencies, and its long-term forecasting results on Udata are worse than other baselines, indicating the importance of prior knowledge on long-term forecasting. Third, the China delay dataset is significantly more challenging than the U.S. delay dataset for deep learning methods, with Gwave not performing well due to its structure not processing missing data well. STPN still outperforms other baselines on long-range forecasting, highlighting the advantage of the attention mechanism on modeling long-range temporal dependencies. Surprisingly, VAR performs reasonably well despite its simplicity on the

\(^1\)https://github.com/FraLuca/STSGCN
\(^2\)https://pytorch.org/
China delay dataset, indicating that a relatively simple model might be less vulnerable to missing data. Finally, LSTM does not perform well on both the U.S. and China delay datasets, with its performance even worse than VAR’s, demonstrating that pure temporal models are unsuitable for handling long-range temporal dependencies within airport flight delays.

Fig. 3(a) gives the spatial visualization of 3-step ahead arrival delay prediction on U.S. dataset. It is clear that STPN model produces the closest estimation toward true values compared with STGCN and Gwave. Furthermore, STPN better approximates the high arrival delay at Northwestern airports with the learned long-range temporal dependencies. Fig. 3(b) shows the spatial visualization of 12-step ahead (6 hours) arrival delay prediction on China dataset. STPN accurately predicts the high arrival delay at HHA airport (Central red value) 6 hours ahead, while other models fail. To qualitatively illustrate the performance of STPN on multi-step ahead prediction, we also visualize the temporal results in Fig. 3(c) and 3(d). Although all models fail to estimate the sudden rise of arrival delay at ORD airport in Fig. 3(c), STPN still gives the best results compared with other baselines. The missing data ratio of HGH departure delay in Fig. 3(d) is relatively high. However, STPN can still give the most accurate results compared with other baselines. Using those examples, we demonstrate that STPN can be potentially used in practice to perform air traffic flow control [43], given the ability to predict arrival/departure delays several hours ahead.

E. Visualization of Temporal Attention Matrix

The main STPN module for capturing temporal dependencies are self-attention layers, which assign a pairwise attention score between every two points. Although it is wrong to equate attention scores with explanation [44], it can offer plausible and meaningful interpretations [45, 46]. To understand how STPN learns temporal dependencies from delay datasets, we calculate each layer’s attention scores between every time point of the day. We visualize parts of raw attention scores $A^{(l)}_{ij} = \frac{\text{softmax}(Q^{(l)}(K^{(l)})^\top)}{\sqrt{c_k}}$ in Fig. 4. Crucially, every attention matrix in different layers learns distinct temporal attention patterns, reflecting that delays in an airport network contain very complex temporal dependencies. Another observation is that we learn more sparse attention scores as we go deeper into the model. For instance, the second adjacency matrix of the last layer of STPN trained on the China dataset only gives high scores between time points with 2.5-hour delays. This pattern resonates with the fact that the travel time of most Chinese flights ranges from 1.5 hours to 2.5 hours.
Fig. 3. Qualitative visualization of prediction results.

(a) Spatial visualization of 3-step ahead arrival delay prediction on U.S. dataset
(b) Spatial visualization of 12-step ahead arrival delay prediction on China dataset
(c) Temporal visualization of arrival delay prediction on ORD airport from U.S dataset
(d) Temporal visualization of departure delay prediction on HGH airport from China dataset

Fig. 4. Raw Attention maps of STPN at different layers.

(a) Temporal Attention Matrices of STPN trained on U.S. dataset
(b) Temporal Attention Matrices of STPN trained on China dataset

It seems that every temporal attention in deeper layers is only responsible for one type of temporal dependency. We also find that most attention maps in the deeper layers only give high attention scores between temporally closer time points. Only the attention maps in the first layer give uniform attention scores between every time point. It means that the self-attention model of the first STPN layer learns some global temporal dependencies, while deeper layers learn local dependencies. For example, the first adjacency matrix of the first layer gives similar scores for all time points, showing that it learns global correlation. It shows that STPN uses both short-range and long-range temporal dependencies for delay prediction.

F. Ablation Analysis

To understand the importance of the modules in STPN, we consider eight ablations, and evaluate them on the U.S. delay dataset since it contains less missing data:
1) **STPN-WS**: We remove the SE-block in each layer of STPN.

2) **STPN-WC**: We remove the covariate (weather condition) inputs and their associated embeddings.

3) **STPN-WOD**: We remove $A_{D→O}$ and $A_{O→D}$, and only use the distance adjacency matrix $A_d$ to perform graph convolution.

4) **STPN-WDis**: We remove the distance adjacency matrix $A_{Dis}$.

5) **STPN-WG**: We remove all the diffusion graph convolution blocks, which means that the spatial dependencies are not considered.

6) **STPN-S**: Departure and arrival delays are trained separately, so the departure-arrival delay relationship is not considered.

7) **STPN-TCN**: We replace the self-attention mechanism in STPN with a temporal convolutional network (TCN). The kernel size of the TCN is set to 3 so that only nearby time slots are highly correlated. The output layer consists of a linear projection layer in the temporal dimension and a linear projection in the feature dimension.

8) **STPN-LSTM**: We replace the self-attention mechanism in STPN with an LSTM block. In this case, the GCN block in (8) is used to update the gates of LSTM. This model can be regarded as a variant of DCRNN [17], in which different graph convolution and additional SE blocks are used.

The results in Table IV show the performance of these ablations. The best indexes except the ones of STPN are marked in bold, and the worst ones are marked with underlines.

### Table IV

| Method        | overall metrics | Arrival delay | Departure Delay |
|---------------|----------------|---------------|-----------------|
|               | MAE  | RMSE  | R²     | MAE  | RMSE  | R²     |
| STPN          | 6.428| 8.967 | 0.356 | 7.506| 10.165| 0.293 |
| STPN-WS       | 6.456| 9.017 | 0.349 | 7.506| 10.193| 0.289 |
| STPN-WC       | 6.457| 8.996 | 0.352 | 7.523| 10.185| 0.290 |
| STPN-WOD      | 6.448| 9.004 | 0.350 | 7.508| 10.197| 0.289 |
| STPN-WDis     | 6.442| 9.007 | 0.350 | 7.507| 10.188| 0.290 |
| STPN-WG       | 6.487| 9.065 | 0.342 | 7.559| 10.253| 0.281 |
| STPN-S        | 6.521| 9.126 | 0.333 | 7.650| 10.392| 0.261 |
| STPN-TCN      | 6.529| 9.124 | 0.336 | 7.632| 10.320| 0.271 |
| STPN-LSTM     | 6.450| 8.971 | 0.353 | 7.515| 10.198| 0.289 |

**Improvements in multi-graph convolution**: The performance of STPN-WOD and STPN-WDis is slightly worse than those of STPN, indicating that STPN can use one type of relation to infer sufficient spatial dependencies. However, STPN-WG performs badly in predicting departure delay, suggesting that spatial dependencies are more helpful in predicting departure delay. It is reasonable since the spatial departure delays are more likely local-correlated due to extreme weather conditions.

**Improvements in co-training**: Unsurprisingly, STPN-S gives bad overall performance and arrival delay prediction performance. It is because arrival delay is highly related to departure delay. In many cases, departure delays are the cause of arrival delays. However, the impacts of arrival delays on departure delays are relatively small. Thus the departure delay prediction performance of STPN-S is even better than those of STPN-WG and STPN-WS.

**Improvements in self-attention for temporal modeling**: To justify our choice of self-attention for temporal modeling, we replace the self-attention with TCN and LSTM with a similar parameter size. In Table IV, the errors increase for both STPN-TCN and STPN-LSTM, showing self-attention is an effective choice for the model architecture of delay time series. In particular, STPN-LSTM is better than STPN-TCN. Its overall $RMSE$ and $R^2$ are close to the original STPN. It is because TCN in each layer only uses information from nearby time slots, while airport delay shows strong long-term dependencies. Although the performance of STPN-LSTM also shows solid performance, we still recommend using self-attention. Because it is faster under the same parameter size as all the inputs are ingested once.

**Number of temporal attention heads**: Assuming that the delay of multiple airports contains complex temporal dynamics, we use multiple attention heads to infer temporal adjacency matrices. We explore STPN with different numbers of temporal attention heads to judge the contribution of multiple temporal adjacency matrices. Results on U.S. dataset are shown in Fig. 5. All models in Fig. 5 contain four layers of space-time-separable multi-graph convolution. The results show that the optimal number of attention heads is 4, slightly better than the one that only learns one temporal adjacency matrix in each layer. However, more heads induce difficulty in training, leading to increased delay patterns, and STPN can almost learn those effects by itself.
errors. In Fig. 5, the models with 8 and 16 temporal adjacency matrices give higher errors than the ones with four heads. It should be noted that STPN learns different temporal attention matrices in different layers. Therefore the model with one head also learns four types of temporal dependencies in this ablative experiment.

Number of layers: The number of layers has a nonnegligible impact on the model’s accuracy. With more layers, STPN can theoretically extract more complex ST dependencies. However, training deeper graph convolutional networks is still challenging due to the over-smoothing issue [47, 48].

We test two-layer STPN with 128 hidden neurons, three-layer STPN with [128, 64] hidden neurons, four-layer STPN with [128, 64, 32] hidden neurons, five-layer STPN with [128, 64, 32, 32] hidden neurons, and six-layer STPN with [128, 64, 32, 32, 32] hidden neurons on U.S. dataset. Fig. 6 shows that the four-layer model outperforms all other models. Our intuition is that this setting allows STPN to be trained easily due to its shallowness, and the four-layer model can extract more meaningful ST delay propagation patterns than too shallow structures. However, we have also only performed this ablation study on a relatively shallow structure, so it might be the case that deeper architectures perform better. There is lots of room for future exploration in this design choice.

G. Counterfactual Intervention

The ultimate goal of the delay prediction model is to guide intervention for delay reduction. Therefore, the prediction model capable of guiding intervention should be able to think about the impact of alternatives to reality (counterfactuals) [49]. The counterfactual ability allows human users to simulate some aspects of the targeted system. Specifically, we are particularly interested in counterfactuals about future delay reduction. Reducing delay is a complex task that involves multiple controls of various degrees of granularity [50]. In this work, we explore a specific intervention: we assume that we can reduce the historical departure delay of the busiest airports to 0 s. This is reasonable since departure delays can be seen as the origin of all delays. After the intervention, we examine the delay reduction before and after intervention based on \( \Delta x = x_{\text{pred}} - \hat{x}_{\text{pred}} \), in which \( x_{\text{pred}} \) is the normal prediction with true input, and \( \hat{x}_{\text{pred}} \) is the prediction with intervened inputs.

We depict an intervention example in Fig. 7. In Fig. 7, large-scale departure delays happened in the U.S. airport network. We reduce the historical departure delay of busy airports ATL, LAX, and DIA to 0 s. The following observations should be highlighted from Fig. 7: (1) The intervention dramatically reduces the departure and arrival delays of the intervened airport in a short-term temporal range. The arrival and departure delay reductions were larger than 8 minutes from 16:00 to 18:00. As expected, the intervention, which reduces the departure delay, can increase the capacity of airports. As a result, it can dramatically reduce future arrival and departure delays. After a short period (18:00 in Fig. 7(a)), the intervention effects dropped. The reason might be that the flight delay from other airports will eventually cascade to ATL and mitigate the effects of the intervention. (2) The intervention can also dramatically reduce the arrival delays of parts airports. Fig. 7(a) gives an example of PDX airport, whose arrival delay has been significantly influenced by the intervention. The arrival delay is dropped by 4 minutes, while the departure delay of PDX has not been greatly affected. It means that if an aircraft reduces its departure delay, this reduced departure delay may directly reduce the arrival delay, and it can indirectly propagates to departure delay. We also observe the reduction of arrival delay drops, and the one of departure delay rises after 18:00. The shortest travel time from DIA and LAX to PDX ranges from 2 to 2.5 hours. After 2.5 hours, only the intervened flights from ATL could directly impact arrival delay. The results show that STPN has learned this physical relationship. (3) The local delay reduction can propagate through part or all of the airport network. In Fig. 7(b), the intervention nearly causes a great delay reduction in all Northwestern airports. After 6 hours, we also observed a large reduction of arrival delays in the intervened airports. The more times the intervened aircraft takes off from intervened airports, the more susceptible it becomes to upstream delay reduction that may affect subsequent visits to intervened airports. In a nutshell, reduced departure delay of the intervened airport causes high arrival delay reduction onto the intervened airport itself. Compared with arrival delay, the intervention only has local effects on departure delay. In Fig. 7(c), the reductions of departure delays are far smaller than those of arrival delays in
Fig. 7. Historical departure delay intervention results.

(a) Average departure delay of the historical inputs, and predicted delay reduction of ATL and PDX

(b). Predicted arrival delay reduction

(c). Predicted departure delay reduction

Fig. 7(b), except for the intervened airports. In summary, those counterfactual observations show that our STPN has learned some realistic delay propagation patterns from the dataset. We aim to investigate this subject more carefully in future work.

V. EXTENSION TO OTHER ST FORECASTING PROBLEMS

STPN is capable of capturing ST dependencies across multiple modes, making it applicable to a wide range of general ST prediction tasks. In order to test the generalization capability of ST prediction models, we evaluated STPN’s performance on publicly available traffic flow datasets (PeMS03, 04, 07 and 08 [51]), OD datasets (CDP and SLD [53]), and bicycle datasets (UCTB NYC, Chicago and DC [54]). For traffic flow and OD datasets, we compared STPN’s performance with that of existing works reported in the literature. For UCTB bicycle dataset, we reran the experiments using 80% of the data as training data and the last 10% as test data. We used the adjacency matrices reported in the literature for STPN, with the exception of the PeMS datasets where we used the correlation between daily trend and daily trend difference as two additional adjacency matrices. The results are reported in Tables V, VI and VII.

Overall, STPN showed competitive performance across various ST traffic datasets with different types, distributions, and ST granularities. It achieved the best performance on PeMS03 and PeMS07 and outperformed many existing GNN baselines on PeMS04 and PeMS08. Unlike airport delay patterns, traffic flow is heavily influenced by unknown factors such as travel and driving behaviors. The inclusion of a self-learning spatial adjacency matrix block, such as AGCRN [55] and Gwave [32], could further improve its performance, although this topic is not within the scope of this paper. The results of STPN on OD flow data are very encouraging. It significantly outperforms STZINB in terms of true-zero rate, even though STZINB uses an additional zero-inflated block to identify zero values. For STPN, we trained it using RMSE loss and rounded the predictions to integers. The results in Table VI demonstrate that STPN can generalize well to sparse ST data.

Among all the baselines, STPN achieved the best performance on the UCTB bicycle datasets at a temporal resolution of 1 h. However, its advantage over other baselines became weaker under 30-minute and 15-minute resolution cases. We speculate that this may be due to the correlation and interaction adjacency matrices provided by UCTB [54] being computed at a global level, and their importance becoming weaker under high time resolutions. It is important to note that we only provided a simple evaluation of STPN on these datasets, and further studies are needed to extend STPN-like concepts and structures to these ST datasets.
TABLE V
Performance Comparison of Different Approaches on the PeMS Traffic Flow Datasets. The Results of Other GNN Baselines are from [51], [52].

| Methods | Datasets | Metrics | SVR | LSTM | STGCN | Gwae | STFGNN | AGCRN | STPN |
|---------|----------|---------|-----|------|-------|------|--------|-------|------|
| PeMS03  | MAE      | 21.07   | 21.33 | 17.49 | 19.85 | 16.77 | 15.98  | 15.74 |
|         | MAPE     | 21.07   | 23.33 | 17.15 | 19.31 | 16.30 | 15.23  | 16.40 |
|         | RMSE     | 35.29   | 35.11 | 30.12 | 19.31 | 28.34 | 28.25  | 24.63 |
| PeMS04  | MAE      | 28.70   | 27.14 | 22.70 | 25.45 | 19.83 | 19.83  | 20.38 |
|         | MAPE     | 19.20   | 18.20 | 14.59 | 17.29 | 13.02 | 12.97  | 13.28 |
|         | RMSE     | 44.56   | 41.59 | 35.55 | 39.70 | 31.88 | 32.30  | 32.52 |
| PeMS07  | MAE      | 32.49   | 29.98 | 25.38 | 26.85 | 22.07 | 22.37  | 22.29 |
|         | MAPE     | 14.26   | 13.20 | 11.08 | 12.12 | 9.21  | 9.21   | 9.61  |
|         | RMSE     | 50.22   | 45.84 | 38.78 | 42.78 | 35.80 | 36.55  | 34.97 |
| PeMS08  | MAE      | 23.25   | 22.20 | 18.02 | 19.13 | 16.64 | 15.95  | 16.50 |
|         | MAPE     | 14.64   | 14.20 | 11.40 | 12.68 | 10.60 | 10.09  | 10.54 |
|         | RMSE     | 36.16   | 34.06 | 27.83 | 31.05 | 26.22 | 25.22  | 25.90 |

TABLE VI
Model Comparison for OD Flow Prediction Under Different Metrics. The Results of Other GNN Baselines are from [53].

| Datasets       | Metrics          | STPN | STZINB | HA | STGCN |
|----------------|------------------|------|--------|----|-------|
| CDP-Samp10     | MAE              | 0.351| 0.368  | 0.522| 0.395 |
|                | True-zero rate   | 0.969| 0.796  | 0.759| 0.800 |
|                | F1-Score         | 0.980| 0.848  | 0.809| 0.840 |
| SLD-15min      | MAE              | 0.356| 0.370  | 0.418| 0.373 |
|                | True-zero rate   | 0.900| 0.725  | 0.703| 0.708 |
|                | F1-Score         | 0.903| 0.751  | 0.744| 0.750 |

TABLE VII
RMSEs of Different Approaches on the Bikesharing Datasets. The Datasets are from [54].

| Methods | Datasets | Metrics | STGCN | Gwae | STFGNN | STPN |
|---------|----------|---------|-------|------|--------|------|
|         | 1 hour   | 3.773   | 3.036 | 3.411| 2.871  |
|         | 30 min   | 2.286   | 1.999 | 2.188| 1.938  |
|         | 15 min   | 1.661   | 1.494 | 1.692| 1.471  |
|         | 1 hour   | 2.910   | 3.158 | 2.978| 2.829  |
|         | 30 min   | 1.606   | 1.766 | 1.635| 1.618  |
|         | 15 min   | 1.112   | 1.059 | 1.135| 1.093  |
|         | 1 hour   | 2.637   | 2.439 | 2.489| 2.216  |
|         | 30 min   | 1.484   | 1.425 | 1.423| 1.407  |
|         | 15 min   | 1.089   | 0.994 | 1.123| 1.012  |

VI. CONCLUSION AND FUTURE WORKS

We have presented a deep learning model named STPN for predicting multi-step arrival and departure delay in large networks of airports, taking into account the spatiotemporal dependencies, exogenous factor effects, and departure-arrival delay relationship. Because of the efficiencies gained through the space-time-separable multi-graph convolution framework we have adopted, as well as several specific structure designs for spatiotemporal dependencies mining, the STPN provides more accurate predictions compared with several baselines. Moreover, the counterfactuals produced by STPN can be a powerful tool for exploring, from a macroscopic perspective, the implications of a broad range of alternative strategies concerning delay reduction interventions in a regional or national system of airports.

Delay propagation modeling is a central notion in modern air traffic management systems, which aims to exploit the compact spatiotemporal dependencies underlying real-world aviation systems. Granted, the possibilities for future research from this paper are extensive. From the perspective of deep neural networks for delay propagation learning, ongoing work involves more advanced structure design. A dynamic extension of this work’s static sum-of-Kronecker spatiotemporal kernel would be desirable. Further extensions could include the structure for handling missing data. It would also be interesting to explore how effectively the model can be used for small airports with sparse flights. In the aspects of real-world applications, the STPN only works from the macroscopic perspective. Developing models for learning delay propagation between different individual aircraft at the mesoscopic level would also be necessary. To that end, the work presented in this paper suggests new insights on data-driven delay propagation learning that may benefit air traffic management.

REFERENCES

[1] International Air Transport Association (IATA), “Annual review 2019,” Accessed: Feb. 2020. [Online]. Available: https://www.iata.org/contentassets/
[2] N. de Loisy, Transportation and the Belt and Road Initiative. Supply Chain Management Outsource Limited, Seattle, WC, USA: Amazon Digital Services LLC, 2019.
[3] D. Bertsimas, G. Lulli, and A. Odoni, “An integer optimization approach to large-scale air traffic flow management,” Operations Res., vol. 59, no. 1, pp. 211–227, 2011.
[4] D. Bertsimas and S. Gupta, “Fairness and collaboration in network air traffic flow management: An optimization approach,” Transp. Sci., vol. 50, no. 1, pp. 57–76, 2016.
[5] S. Ayan and H. Samet, “Aircraft trajectory prediction made easy with predictive analytics,” in Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining, 2016, pp. 21–30.
[6] N. Kafle and B. Zou, “Modeling flight delay propagation: A new analytical-econometric approach,” Transp. Res. Part B: Methodological, vol. 93, pp. 520–542, 2016.
[7] M. Z. Li, K. Gopalakrishnan, K. Pantoyo, and H. Balakrishnan, “Graph signal processing techniques for analyzing aviation disruptions,” Transp. Sci., vol. 55, no. 3, pp. 553–573, 2021.
[1] Q. Cai, S. Alam, and V. N. Duong, “A spatial–temporal network perspective for the propagation dynamics of air traffic delays,” *Engineering*, vol. 7, no. 4, pp. 452–464, 2021.

[2] N. Pyrgiotis, K. M. Malone, and A. Odoni, “Modelling delay propagation within an airport network,” *Transp. Res. Part C: Emerg. Technol.*, vol. 27, pp. 60–75, 2013.

[3] I. Simiaikis and H. Balakrishnan, “A queuing model of the airport departure process,” *Transp. Sci.*, vol. 50, no. 1, pp. 94–109, 2016.

[4] J. J. Rebollo and H. Balakrishnan, “Characterization and prediction of air traffic delays,” *Transp. Res. Part C: Emerg. Technol.*, vol. 44, pp. 231–241, 2014.

[5] B. Yu, Z. Guo, S. Asian, H. Wang, and G. Chen, “Flight delay prediction for commercial air transport: A deep learning approach,” *Transp. Res. Part E: Logistics Technol.*, vol. 125, pp. 203–221, 2019.

[6] J. Bao, Z. Yang, and W. Zeng, “Graph to sequence learning with attention mechanism for network-wide multi-step-ahead flight delay prediction,” *Transp. Res. Part C: Emerg. Technol.*, vol. 130, pp. 103323, 2021.

[7] K. Cai, Y. Li, Y.-P. Fang, and Y. Zhu, “A deep learning approach for flight delay prediction through time-evolving graphs,” *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 8, pp. 11397–11407, Aug. 2021.

[8] T. Sofianos, A. Sampieri, L. Franco, and F. Galasso, “Space-time-separable graph convolutional neural network for pose forecasting,” in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 11209–11218.

[9] A. Vavaspardis et al., “Attention is all you need,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.

[10] Y. Li, R. Yu, C. Shahabi, and Y. Liu, “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting,” in *Proc. Int. Conf. Learn. Representations*, 2018.

[11] J. Hu, L. Shen, and G. Sun, “Squeeze-and-excitation networks,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7132–7141.

[12] R. Beatty, R. Hsu, L. Berry, and J. Rome, “Preliminary evaluation of flight delay propagation through an airline schedule,” *Air Traffic Control Quart.*, vol. 7, no. 4, pp. 259–270, 1999.

[13] P. Fleurquin, J. J. Ramasco, and V. M. Eguiluz, “Systemic delay propagation in the us airport network,” *Sci. Rep.*, vol. 3, no. 1, pp. 1–6, 2013.

[14] B. Campanelli et al., “Comparing the modeling of delay propagation in the us and European air traffic networks,” *J. Air Transport Manage.*, vol. 56, pp. 12–18, 2016.

[15] C.-L. Wu and K. Law, “Modelling the delay propagation effects of multiple resource connections in an airline network using a Bayesian network model,” *Transp. Res. Part E: Logistics Technol.*, vol. 122, pp. 62–77, 2019.

[16] M. Liu et al., “SCINet: Time series modeling and forecasting with sample convolution and interaction,” in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 3538–3545.

[17] K. Xu, C. Li, Y. Tian, T. Sonobe, K.-I. Kawarabayashi, and S. Jegelka, “Representation learning on graphs with jumping knowledge networks,” in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2018, pp. 5453–5462.

[18] R. M. J. Byrne, “Counterfactuals in explainable artificial intelligence (XAI): Evidence from human reasoning,” in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 6276–6282.

[19] A. K. Agogino and K. Tumer, “A multigagent approach to managing air traffic flow,” *Auton. Agents Multi-Agent Syst.*, vol. 24, no. 1, pp. 1–25, 2012.

[20] M. Li and Z. Zhu, “Spatial-temporal fusion graph neural networks for traffic flow forecasting,” in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 4189–4196.

[21] M. Liu et al., “SCINet: Time series modeling and forecasting with sample convolution and interaction,” in *Proc. 26th Conf. Neural Inf. Process. Syst.*, 2022, pp. 5816–5828.

[22] D. Zhang, S. Wang, H. Koutsopoulos, and J. Zhao, “Uncertainty quantification of sparse travel demand prediction with spatial-temporal graph neural networks,” in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 4639–4647.

[23] L. Wang, D. Chai, X. Liu, L. Chen, and K. Chen, “Exploring the generalizability of spatio-temporal traffic prediction: Meta-modeling and an analytic framework,” *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 4, pp. 3870–3884, Apr. 2021.

[24] L. Bai, L. Yao, C. Li, X. Wang, and C. Wang, “Adaptive graph convolutional recurrent network for traffic forecasting,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 17804–17815.
Hongyu Yang received the BS degree from Shanghai Jiaotong University, in 1998, and the MS and PhD degrees from Sichuan University, in 1991 and 2008, respectively. She is currently a professor with the College of Computer Science, Sichuan University, China. Her research interests include visual synthesis, digital image processing, computer graphics, real-time software engineering, air traffic information intelligent processing technology, visual navigation, and flight simulation technology.

Yi Lin (Member, IEEE) received the PhD degree from Sichuan University, Chengdu, China, in 2019. He currently works as an associate professor with the College of Computer Science, Sichuan University. He was also a visiting scholar with the Department of Civil and Environmental Engineering, University of Wisconsin–Madison, Madison, WI, USA. His research interests include air traffic flow management and planning, machine learning, and deep learning-based air traffic management applications.

Hong Liu received the PhD degree from Sichuan University, Chengdu, China, in 2014. He currently works as an associate professor with the College of Computer Science, Sichuan University. His research interests include intelligent transportation systems, and intelligent decision making.