Contents lists available at ScienceDirect

Journal of King Saud University – Computer and Information Sciences

journal homepage: www.sciencedirect.com

Journal of King Saud University – Computer and Information Sciences

Detecting anomalous traffic behaviors with seasonal deep Kalman filter graph convolutional neural networks

Yanshen Sun [a,*], Yen-Cheng Lu [a], Kaiqun Fu [b], Fanglan Chen [a], Chang-Tien Lu [a]

[a] *Department of Computer Science, Virginia Tech, VA, USA*
[b] *Department of Computer Science, South Dakota State University, SD, USA*

ARTICLE INFO

Article history:
Received 20 January 2022
Revised 25 May 2022
Accepted 26 May 2022
Available online xxxx

Keywords:
Traffic forecasting
Spatiotemporal fusion
Multi-granular seasonal feature
Graph neural network
Anomaly detection

ABSTRACT

Anomaly detection over traffic data is crucial for transportation management and abnormal behavior identification. An anomaly in real-world scenarios usually causes abnormal observations for multiple detectors in an extended period. However, existing anomaly detection methods overly leverage the single or isolated feature interdependent contextual information in anomalies, inevitably dropping the detection performance. In this paper, we propose S-DKFN (Seasonal Deep Kalman Filter Network), to identify abnormal patterns with a long duration and wide coverage. S-DKFN models traffic data with a graph and simultaneously investigates the spatial and temporal features to hunt abnormal behaviors. Specifically, a dilation temporal convolutional network (TCN) is used to merge the multi-granular seasonal features and a graph convolution network (GCN) to extract spatial features. The outputs of TCN and GCN are then fed to long-short term models (LSTM) and merged by Kalman filters for denoising. An encoder-decoder module is introduced to predict traffic attributes with seasonal features. The mean squared errors (MSE) of the predictions are considered the anomaly scores. Experimental results on two real-world datasets show that our proposed S-DKFN framework outperforms the state-of-the-art baseline methods in detecting anomalies with long-duration and wide-coverage, especially its ability to detect accidents.

1. Introduction

Intelligent transportation systems (ITS) aim to improve the level of utilization of road networks under the limited traffic infrastructures, which focuses on relieving traffic congestion and reducing the risk of traffic incidents. Numerous spatiotemporal traffic data are being produced every day with the growing number of wireless sensors, which provides the raw material for tracking the traffic status and mining significant insights about abnormal behaviors such as traffic jams and accidents. Recently, an increasing number of data-driven based applications have been developed, including traffic monitoring (Barbagli et al., 2011), traffic flow forecasting (Guo et al., 2019; Yu et al., 2021), traffic routing (Megalingam et al., 2011), and traffic anomaly detection (Liu et al., 2011). Indeed, traffic anomaly detection has been becoming the most significant issue due to the fact that traffic anomalies may be the cause of most of the disruptions in freeway traffic flow.

Anomaly detection, in general, aims to identify data instances that exhibit unexpected behaviors. This technique can capture the traffic patterns in large-scale datasets and detect potentially informative or actionable insights that typically remain undiscovered. More specifically, the anomalies in the traffic sensor reporting could signify various underlying issues, such as traffic accidents, signal failures, traffic congestion, sensor failures, and unknown issues that require further investigation (Djenouri et al., 2019). Numerous studies have focused on specific traffic anomalies introduced by occasional activities (Liu et al., 2011; Pang et al., 2021; Pang et al., 2013), and the others have focused on recurring traffic anomalies (Chow et al., 2014). According to a recent survey on methods for detecting traffic anomalies (Djenouri et al., 2019), this field of study is still in its infancy, and additional research utilizing, for example, computational intelligence, optimization, and high-performance computing, is needed.

Traditional traffic modeling methods focus on capturing temporal dependencies. Techniques such as Auto-Regressive Integrated

[*] Corresponding author.

E-mail addresses: yansh93@vt.edu (Y. Sun), kevinlu@vt.edu (Y.-C. Lu), Kaiqun.Fu@sdstate.edu (K. Fu), fanglanc@vt.edu (F. Chen), clu@vt.edu (C.-T. Lu).

Peer review under responsibility of King Saud University.

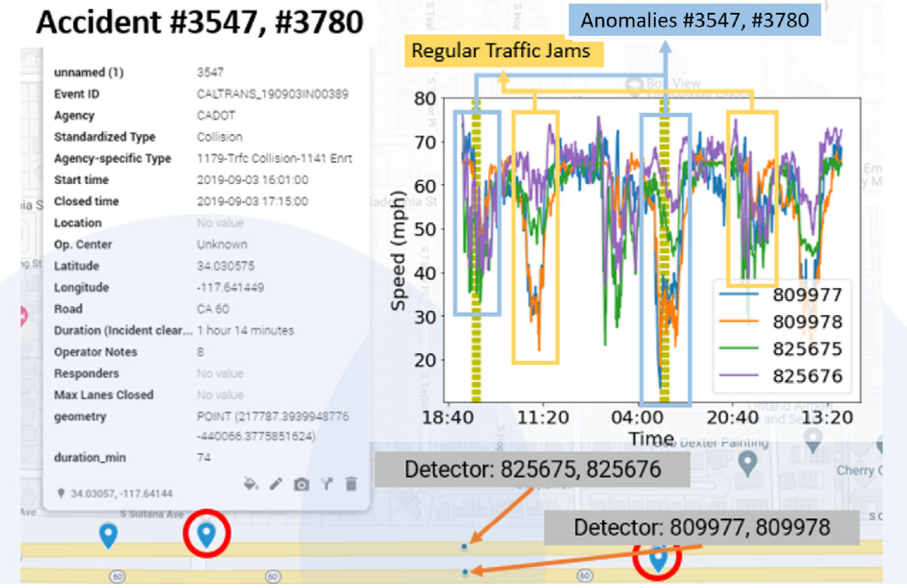
ELSEVIER Production and hosting by Elsevier

https://doi.org/10.1016/j.jksuci.2022.05.017

Moving Average (ARIMA) (Ahmed and Cook, 1979), Support Vector Regression (SVR) (Smola and Schölkopf, 2004), and deep learning based approaches like Recurrent Neural Networks (RNNs) (Rumelhart et al., 1985) and its modified version, Gated recurrent units (GRU) (Cho et al., 2014) and long short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) have been widely applied with their inherent strengths in modeling time series. However, none of these methods can handle the intrinsic geographic dependencies in traffic data.Studies have extended the idea of convolutional networks to graph-structured data by applying the convolution operators over the edges and nodes of the graphs (Kipf and Welling, 2017). Yang et al. (2020) presented a robust graph convolutional network (GCN)-based approach for person re-identification in videos. Valada and Burgard (2017) proposed a robustness-aware training approach for terrain detection. Ma et al. (2021) provided a comprehensive review of the deep learning techniques for graph anomaly detection. These approaches can be further applied to modeling spatial dependencies, and thus becomes a valuable solution for capturing the spatiotemporal patterns in traffic data. Recently, spatiotemporal modeling techniques have attracted attention in the traffic and transportation research domain. Many spatiotemporal modeling approaches have been presented, but most of them are not designed to handle the large variances that may be introduced by traffic anomalies. As shown in 1, distinguishing traffic incidents from regular jams is a challenging task as the features exhibited by jams and accidents usually show similar patterns. It is obvious that the locations of the accident #3547 on the left and #3780 on the bottom and the speeds of traffic jam and accident both lead to drops. Therefore, it is difficult for traditional methods that only consider speed features to distinguish the differences between jams and accidents, that is, due to masking and swamping effects (Bendre, 1989), anomalies can often affect the fitting process of a model, thus causing it to capture a biased pattern.

The challenges of anomaly detection in traffic data include the following: (1) *Modeling spatiotemporal dependencies and capturing abnormal patterns.* Most of the existing anomaly detection models focus on temporal relations while ignoring the spatial patterns of anomalies, resulting in a large amount of valuable spatial information lost. For example, an incident (within the light blue circle in Fig. 1) often triggers multiple neighbour detector anomalies, whose pattern changes can provide more comprehensive information for anomaly detection. (2) *Distinguishing implicit anomalies from regular variation.* Anomalies may lead to feature patterns similar to those of regular variations, while their contextual modes are different. For example, for detector #825675 in Fig. 1, a drop in speed at around 9 pm is most likely a regular variation of speed, while a low speed at around 11 pm may indicate an accident. This challenge makes it especially important to capture patterns while taking spatiotemporal dependencies and traffic connectivity into account. (3) *Maintaining robustness with a lack of normal references.* A traffic accident may affect the speeds measured by several adjacent detectors for an extended period. As shown in Fig. 1 accident #3547 lasts for two hours and covers all the neighbors of detector #825675. The predicted value of detector #825675 could be very close to the abnormal value if only the previous one hour's data are used for prediction. Existing studies are based on either temporal only modeling methods, or spatiotemporal modeling approaches based on an Euclidean assumption, which ignores the nature that the traffic of a specific road section is often affected more by the connectivity than the Euclidean distances to the other road sections (4) *Complex internal structures.* Traffic anomaly data is typically characterized by several complex internal structures such as trend, seasonality, stationarity, and auto-correlation. The internal structures of the data require special formulation and techniques for their analysis. It is essential to develop effective models to explore the interdependent relationships among seasonal, cyclical, and irregular components in the traffic data. These models are then used to predict the series for future points in time.

To overcome the challenges above, we propose the seasonal deep Kalman filter network (S-DKFN), a novel spatiotemporal anomaly detection method based on graph neural network. The model leverages multi-scale heterogeneous features to distinguish anomalies from their contexts. The denoised combination of spatial and temporal information ensures the recovery of normal patterns with little normal information. The significant contributions of this paper are summarized below.



Accident #3547, #3780

unnamed (1)	3547
Event ID	CALTRANS_190903IN00389
Agency	CADOT
Standardized Type	Collision
Agency-specific Type	1179-Trfc Collision-1141 Enrt
Start time	2019-09-03 16:01:00
Closed time	2019-09-03 17:15:00
Location	No value
Op. Center	Unknown
Latitude	34.030575
Longitude	-117.641449
Road	CA 60
Duration (Incident clear...	1 hour 14 minutes
Operator Notes	8
Responders	No value
Max Lanes Closed	No value
geometry	POINT (217787.3939948776 -440066.3775851624)
duration_min	74

34.03057, -117.64144

Fig. 1. Speed drops of regular traffic congestion against an accident. Accident #3547 and #3780 (blue markers with red circles) affects the speed measurements of several surrounding sensors (blue dots).

- **Proposing S-DKFN, a novel unsupervised traffic anomaly detection model based on temporal and spatial features fusion to hunt more comprehensive anomaly patterns.** The model incorporates features of different spatial and temporal resolutions while maintaining the atomicity of training samples. It leverages different neural network and denoising techniques to produce smooth predicted time series.
- **Proposing a mechanism to incorporate seasonal features through a multi-level sliding window.** Sliding windows turn historical timestamps in larger windows into seasonal features and thus shorten the lengths of training sequences. This method improves the effectiveness of long short-term memory networks (LSTMs) and overcomes the inability of LSTMs to handle nonadjacent timestamps.
- **Designing a dilation temporal convolution networks (TCN) module and an encoder-decoder module.** The TCN module assembles different levels of seasonal features, which ensures that no noise by irrelevant timestamps is introduced. The encoder-decoder module is adopted to expand single-season features to multi-seasonal features, forcing the model to drop some small-scale features to enhance the robustness of S-DKFN.
- **Leveraging the principle concept of the Kalman filter for model merging.** The Kalman filter merges the spatial and temporal features by rectifying the high-noise features with low-noise features. The optimized features and a residual of the unmerged features are used for prediction in the next step.
- **We conduct ample experiments on two real-world datasets to evaluate the effectiveness and efficiency of S-DKFN.** Our proposed model is evaluated on the METR-LA and PEMS08 datasets with anomalies of different durations and coverages. The proposed method outperforms competing methods over multiple metrics. The experimental results verify that S-DKFN outperforms the state-of-the-art methods in detecting long duration and large-coverage anomalies.

The rest of this paper is organized as follows. Section 2 reviews the subject background and related work. Section 3 defines the problem. Section 4 introduces the details of S-DKFN. Section 5 presents experiments on two real traffic datasets and the paper is concluded in Section 6.

2. Related works

In this section, we survey the literature of the related research topics, namely temporal and spatiotemporal modeling, general anomaly detection approaches, and traffic anomaly detection.

2.1. Temporal and spatiotemporal modeling

Time series modeling. Temporal modeling has been extensively researched for decades. Traditional approaches including historical average (HA) (Liu and Guan, 2004), auto-regressive integrated moving average (ARIMA) (Hamed et al., 1995), support vector regression (SVR) (Smola and Schölkopf, 2004), and hidden Markov models (HMMs) (Rabiner, 1990) have been widely applied in various domains. In recent years, neural networks models such as recurrent neural networks (Rumelhart et al., 1985) and their variants, including LSTM (Hochreiter and Schmidhuber, 1997; Zhu and Laptev, 2017), gated recurrent unit (GRU) (Cho et al., 2014), and WaveNet (van den Oord et al., 2016) have also been popular for modeling time series data.

Spatiotemporal modeling. Traditional modeling approaches often base on Euclidean assumptions to capture the spatial dependencies. Bruna et al. first proposed a combination of a spatial method and a spectrum method that generalized convolutional neural networks (CNN) from a Euclidean domain to a non-Euclidean domain (Bruna et al., 2014). Following studies extended this idea and formally introduced GCNs (Kipf and Welling, 2017), which generalize the convolution operator from grid-based data to graph data. Recent literature has proven that GCNs achieve superior performance in various domains (Duvenaud et al., 2015; Battaglia et al., 2016). With the success of GCN, Yu et al. (2018) proposed a hybrid approach to model the spatiotemporal effects with a gated CNN that captures temporal patterns, and a GCN captures the spatial patterns. Guo et al. (2019) enhanced the model of Yu et al. with attention networks. Wu et al. (2019) proposed a combination of WaveNet and GCN for general-purpose spatiotemporal prediction. STAWnet (Tian and Chan, 2021) is one of the newest spatiotemporal forecasting models which utilizes hierarchical temporal features and attention to improve performance.

2.2. Anomaly detection

General anomaly detection. Existing anomaly detection approaches can be sorted into two major categories (Pang et al., 2021), namely anomaly measure-dependent learning and generic feature learning. The anomaly measure-dependent methods focus on measuring a particular anomaly measure. For example, the distance-based methods (Knorr and Ng, 1999; Ramaswamy et al., 2000), one-class classification methods (Moya et al., 1993; Roth, 2005), and clustering-based methods (He et al., 2003) fall in this category. These types of methods are well studied and normally easily to be implemented, but struggles with more complicated distributions within the normal class. The generic feature learning group of methods is based on the models that are not specifically designed for anomaly detection. More recent studies adopt deep learning techniques to identify anomalous objects (Ma et al., 2021). For instance, models based on auto-encoders (Doersch, 2021; Lu et al., 2017) can identify anomalies by poorly reconstructed instances. Approaches based on generative adversarial networks (GAN) (Schlegl et al., 2017; Hundman et al., 2018; Medico, 2020) aim to learn a hidden feature space that can capture the normality, and thus detect the anomalies by the differences between the actual instances and the generated instances. Graph-based deep learning methods are also powerful tools for exploring anomaly communities, as they can create much more powerful representations of node attributes and community structures (Liu et al., 2020; Su et al., 2021). These types of approaches are often used for image or video anomaly detection (Liu et al., 2018; Ye et al., 2019).

Traffic anomaly detection. Traffic anomaly detection has been studied in the field of data mining, urban computing, and transportation. Various approaches to detecting abnormal traffic status have been explored. Liu et al. (2011) proposed an early study to detect spatiotemporal anomaly series based on transforming the GPS trajectory data into a region graph, and identified the anomalies from the attribute distances between each pair of graph links in the same time frame. Tisljaric et al. (2020) presented an approach to detect traffic anomalies in GPS data using tensor decomposition techniques. This approach transforms traffic patterns to a speed transition matrix, and identifies the anomalies based on the significance of the Kullback–Leibler (KL) divergence. Yu et al. (2021) presented a deep spatiotemporal graph convolutional network to predict traffic accidents. Deb and Liew (2019) proposed an algorithm called NoiseCleaner to identify and correct noisy categorical attributes values in large traffic accident datasets to avoid misleading the model by noise. Overall, most of the existing traffic anomaly detection approaches leverage neither graph neural networks nor fixed sensor data. In this paper, we propose a novel method based on a robust spatiotemporal predictive model capable of detecting traffic anomalies in fixed sensor datasets.

3. Problem statement

This section first provides the definitions and notations of traffic networks and seasonal features in our research. Then, Section 3.3 briefly explains how a prediction model works as an unsupervised anomaly detection model.

3.1. Traffic network

In this research, the traffic network is considered as an undirected graph $G = (V, E, A)$, where V is a set of nodes ($|V| = N$) defined by the fixed position traffic detectors. E is the edges connecting the related nodes, and $A \in \mathbb{R}^{N\times N}$ is the adjacency matrix representation of E. The edges between nodes are generated from geodesic or road distances among detectors, based on the assumption that closer nodes are more similar to each other. Each node sensor reports traffic measurements (e.g., average speed, traffic volume, and lane occupancy) at the same frequency. Fig. 2 shows an example of a traffic detector network. The red dots indicate the locations of detectors, which are sensors set at fixed locations along the roads. The lines connecting detectors are the edges between the nodes. The different colors of the edges indicate the various geodesic distances between detector nodes.

3.2. Seasonal features

Given a time series $t = 0, 1, \ldots, T-1, T$. At each timestamp, each of the N detectors in G produces F traffic condition features. The raw data matrix is denoted as $\mathscr{X} = (X_1, X_2, \ldots, X_T)^T \in \mathbb{R}^{T\times N\times F}$. Consider that the current time is t_0 and $\tau *\ p$ timestamps are used for seasonal feature extraction. The seasonal feature is $S_{t_0} = Agg\left(X_{t_0-\tau*\ p+1}, \ldots, X_{t_0-(\tau-1)*\ p+1}, \ldots, X_{t_0-p+1}, \ldots, X_{t_0}\right)^T \in \mathbb{R}^{N\times F}$, where Agg is the employed aggregation function. That is, seasonal features are aggregations of T_p consecutive timestamps every p timestamps.

With different τ and p, the model can incorporate seasonal features with different temporal resolutions. The number of timestamps used to generate seasonal features for one timestamp is $max\{\tau_i\ * p_i\}, i = 0, 1, \ldots, L$, where L is the number of different (τ, p) pairs.

3.3. Unsupervised prediction-based anomaly detection

There are two major tasks in prediction-based anomaly detection models: prediction and anomaly detection. For prediction, the anomaly detection model learns the high-level pattern and generates a series of predicted network features at each timestamp. Then, during anomaly detection, data samples “of a certain distance” from the corresponding predicted values are removed.

3.3.1. Traffic prediction

Note that $X_t \in \mathbb{R}^{N\times F}$ represents the network features at timestamp $t, x_t^v \in \mathbb{R}^F$ is the features of network node v at t, and $x_t^{v,f} \in \mathbb{R}$ is the value of feature f of v at t. We have $X_t = (x_t^1, x_t^2, \ldots, x_t^N)^T$ and $x_t^v = \left(x_t^{v,0}, x_t^{v,1}, \ldots, x_t^{v,F}\right)^T$. With historical features of time sequence $t - T_p + 1, t - T_p + 2, \ldots, t$ and corresponding seasonal features $\{S_{t-T_p+1}, S_{t-T_p+2}, \ldots, S_t\}$ in a time window of length τ, the objective is to estimate features $\left\{x_{t+1}^v, x_{t+2}^v, \ldots, x_{t+T_{pred}}^v\right\}$ of each node v at timestamps $t+1, t+2, \ldots t+T_{pred}$, where T_{pred} is the number of timestamps to be predicted.

3.3.2. Anomaly detection

The distances from the original data samples to the predicted values are estimated through the mean square error (MSE) or the sum of squared errors (SSE). Data samples are considered as “anomalies” if their MSE or SSE values are more significant than a threshold.

4. Methodology

This research turns a sequence of records in a traffic network into anomaly scores in three steps: extracting seasonal features from the original data (Section 4.2), producing predicted results (Section 4.3), and turning the predicted features into anomaly scores (Section 5.4).

4.1. Preliminary

This section provides the theoretical analysis and formula deduction of time series decomposition and the Kalman filter. Time series decomposition is used for seasonal feature extraction, and the Kalman filter is used for merging the outputs of the spatial analysis module and the temporal analysis module. The two methods are used in Facebook-Prophet (F.C.D.S. team, 2017) and DKFN (Chen et al., 2020) separately. Note that this section uses a symbol system independent from the other parts of this paper.

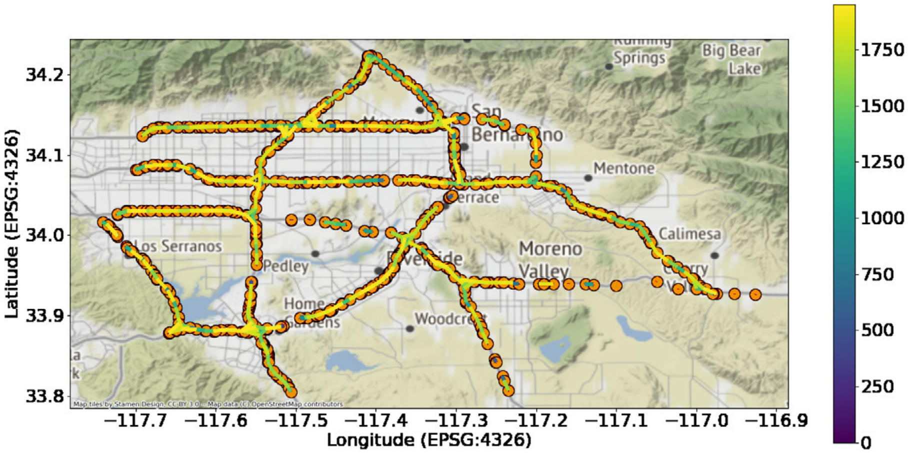


Fig. 2. Traffic detector nodes with edges. The red dots are detectors distributed along roads. The edges are constructed among nodes of geodesic distances within a threshold.

4.1.1. Time series decomposition

Time series decomposition (Enders, 2004) is a technique widely used in "non-neural network" time series analysis models, such as ARIMA (Ahmed and Cook, 1979) and Facebook-Prophet (F.C.D.S. team, 2017). An arbitrary time series Y_t is always composed of a trend series T_t, a seasonal series S_t, and a residual series R_t. The composition method can be either addictive ($Y_t = T_t + S_t + R_t$) or multiplicative ($Y_t = T_t * S_t * R_t$). The additive method will be used as an example below. With a determined period p, T_t at time slot t is computed through a centered moving average $T_t(t) = \frac{1}{p}\sum_i^{[-\frac{p}{2},\frac{p}{2}]} Y_t(i)$. The variation caused by S_t and R_t can then be separated from the original time series by $Y_t - T_t$. S_t is the mean of $\{Y_t - T_t, Y_{t-p} - T_{t-p}, \ldots, Y_{t-k* N_p} - T_{t-k* N_p}\}$, where N_p is the number of period.

Regression-based time series analysis models leverage the whole training dataset for seasonal feature estimation. To maintain the molecular feature of training data samples, in this research, a sliding window of length $Nw_p * p$ is used to compute the seasonal features of the last p time slots, where Nw_p is the number of periods contained in the sliding window.

4.1.2. Kalman filter

The Kalman filter has been frequently used in regression and deep learning models (Coskun et al., 2017; Lu et al., 2018) due to its ability to merge two estimations by minimizing their covariance. The Kalman filter is commonly used for merging the estimation of x_{t+1} from historical data x_t and the measurement z_{t+1} by minimizing the variance of x_{t+1}. Let $x_{t+1} = Ax_t + w$ and $z_{t+1} = Hx_{t+1} + v$, where A and H are coefficient matrices. $w \sim N(0, Q)$ is the prediction noise, and $v \sim N(0, R)$ is the measuring noise. The x_{t+1} predicted from x_t and x_{t+1}'s variance are:

$$\hat{x}'_{t+1} = Ax_t + w \tag{1}$$

$$\widehat{P}'_{t+1} = Ax_tA^T + Q \tag{2}$$

The estimation adjusted by the measurement z_{t+1} can then be written as:

$$\hat{x}_{t+1} = \hat{x}'_{t+1} + K_{t+1}\left(z_{t+1} - H\hat{x}'_{t+1}\right) \tag{3}$$

$$\widehat{P}_{t+1} = (I - K_{t+1}H_{t+1})\widehat{P}'_{t+1} \tag{4}$$

where $K_{t+1} = \widehat{P}'_{t+1}H^T\left(H\widehat{P}'_{t+1}H^T + R\right)^{-1}$ is the Kalman gain. Inspired by Chen et al. (2020), we leverage the principal concept of the Kalman filter for distribution merging. The concepts are described as follows.

For a value x, consider that there are two independent estimations x_1 and x_2, and x is a weighted combination of x_1 and x_2:

$$\hat{x} = w_1x_1 + w_2x_2 \tag{5}$$

Where w_1 and w_2 are weights of x_1 and $x_2, w_1 + w_2 = 1$. The expectation of $\hat{x}, E(\hat{x}) = w_1E(x_1) + w_2E(x_2)$. As $E(x_1)$ and $E(x_2)$ are independent, $E([x_1 - E(x_1)][x_2 - E(x_2)]) = 0$. σ^2 can then be written as:

$$\begin{aligned}\sigma^2 &= E\left([\hat{x} - E(\hat{x})]^2\right)\\ &= w_1^2E\left([x_1 - E(x_1)]^2\right) + w_2^2E\left([x_2 - E(x_2)]^2\right)\\ &= w_1\sigma_1^2 + w_2\sigma_2^2\end{aligned} \tag{6}$$

Let $w_2 = w$ and $w_1 = 1 - w$. To minimize σ^2, let the differential between σ^2 and w be zero:

$$\frac{d\sigma^2}{dw} = -2(1 - w)\sigma_1^2 + 2w\sigma_2^2 = 0 \tag{7}$$

The analytical solution of w is:

$$w = \frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} \tag{8}$$

$\hat{x}$ and σ^2 are:

$$\hat{x} = \frac{\sigma_2^2x_1 + \sigma_1^2x_2}{\sigma_1^2 + \sigma_2^2}, \sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2} \tag{9}$$

In a prediction model, consider that x_1 and x_2 are two estimations generated by two different models. From Eq. 9, it can be inferred that the merged estimation $\hat{x}$ is the weighted sum of x_1 and x_2, where the weights are $\frac{\sigma_2^2}{\sigma_1^2+\sigma_2^2}$ and $\frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}$ respectively.

4.2. Seasonal feature extraction

In traffic networks, attributes, such as speed and occupancy, would not increase or decrease over time. Therefore, the seasonal features S_t are merged with the original time series as the inputs of the time series analysis model.

In recurrent neural networks (RNNs), the input of extended time sequences can lead to extremely long training times and gradient explosion or vanishing. Therefore, a multi-level sliding window method is employed along with time decomposition to extract seasonal features of each time slot. Consider the current time is t_0 and T_p timestamps $t_0 - T_p + 1, t_0 - T_p + 2, \ldots, t_0$ are used for prediction. The base sliding window is of size T_p and the data contained in the sliding window is $\mathscr{X}_0 = (X_{t_0-T_p+1}, X_{t_0-T_p+2}, \ldots, X_{t_0})^T \in \mathbb{R}^{T_p\times N\times F}$.

The first sliding window used for seasonal feature extraction contains $\tau_1 * p_1 + T_p$ timestamps including t_0, that is $\mathscr{X}_1 = (X_{t_0-\tau_1 *p_1-T_p+1}, X_{t_0-\tau_1 *p_1-T_p+2}, \ldots, X_{t_0})^T \in \mathbb{R}^{\tau_1 * p_1+T_p\times N\times F}$. Time series decomposition is performed on $\mathscr{X}_1$ to compute the seasonal features of each element in $\mathscr{X}_1$. Based on the definition of time series decomposition in Section 4.1.1, the first-level seasonal feature $S_{t_0}^1$ of t_0 can be computed as:

$$S_{t_0}^1 = \frac{1}{\tau_1 + 1}\sum(X_t - T_t) \tag{10}$$

where $t = \{t_0 - \tau_1 * p_1, t_0 - (\tau_1 - 1) * p_1, \ldots, t_0\}$ is the T_pth element in every p_1 timestamps. T_t is the trend at timestamp t. In this case, the generated seasonal feature matrix is also of size $T_p \times N \times F$ and the number of features is doubled ($\mathscr{X} \in \mathbb{R}^{(T\times N\times(2 * F)}$).

Other τ and p can be added the same way as τ_1 and p_1. If L windows are used, the number of timestamps used to generate one inT_p and the size of an input sample is $T_p * N * (L * F)$.

4.3. S-DKFN model

As shown in Fig. 3, the S-DKFN model has two components: a spatial analysis module for spatial relation modeling and a temporal analysis module for temporal relation modeling. In the rest of the article, we use GCLSTM to represent the spatial analysis module, as the module consists of GCNs and an LSTM module. TCLSTM is employed to represent the temporal analysis module due to its time convolution network (TCN) and LSTM structure. GCLSTM is used to simulate interactions between each node and its neighbors. Similarly, the "temporal relation" describes historical features' effects on the current timestamp for each node.

At each timestamp, the input of GCLSTM is the adjacency matrix and the original time sequence data, and the input of TCLSTM is the original data and the seasonal features extracted during preprocessing. GCLSTM leverages a GCN block to extract the spatial-dependency features of the input data and feed the

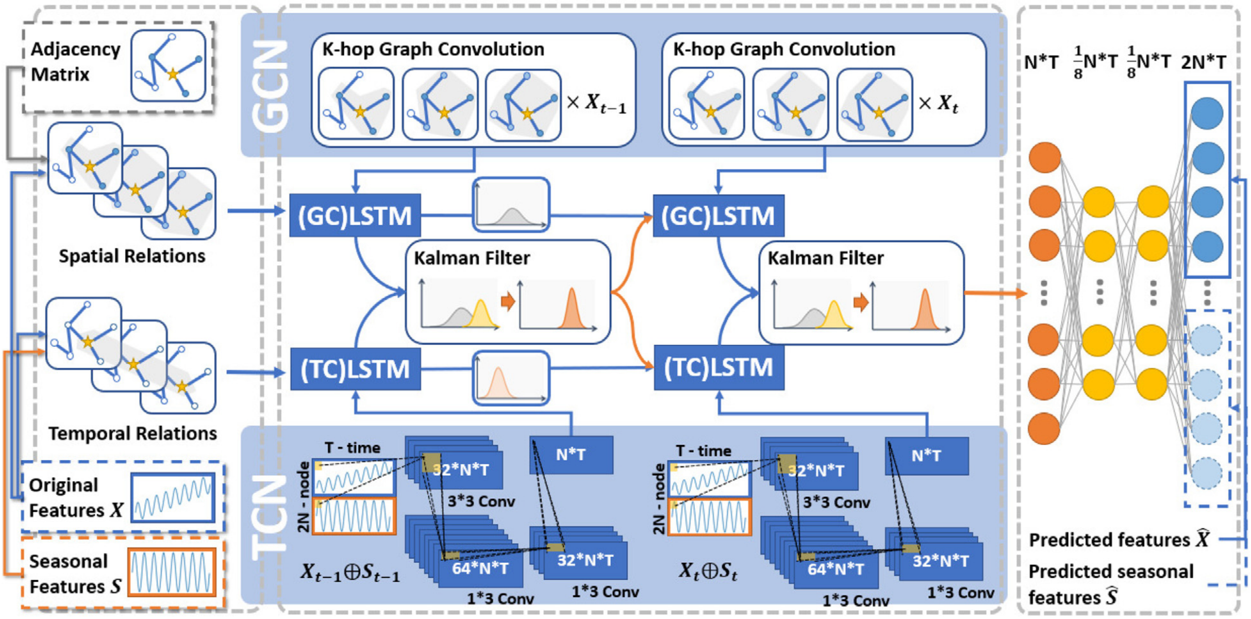


Fig. 3. Network architecture of S-DKFN. For each timestamp, the input of GCN is the k-hop adjacent matrix and X; The input of TCN is X with seasonal features S. The output of the final timestamp is send expanded to $\hat{X}\oplus\hat{S}$ for loss computation.

features to an LSTM layer. In TCLSTM, the original data and the seasonal features are merged with a dilation TCN block and sent to an LSTM layer as well. The outputs of the LSTM layers are merged with the Kalman filter and fed to the next timestamp. At the final timestamp, the outputs of LSTMs are merged with the Kalman filter and then sent to a simple encoder-decoder module to generate a prediction of both the original data and the corresponding seasonal features. The architecture of the model can be found in Fig. 3.

The model considers each feature of the original data separately. To simplify the symbol system, the number of features for each node at each timestamp will be considered as 1 in this section.

4.3.1. GCLSTM

GCN performs convolution on graphs to aggregate information among adjacent nodes. Instead of adjacent pixels, a graph convolutional layer considers linked nodes as "adjacent" and merges the information of each node and their k-hop neighbors with convolution kernels. The classic method is to perform graph convolution in the frequency domain with the Laplacian matrix and learn the k as in "k-hop neighbors" to be aggregated for each node with some approximation method such as Chebyshev polynomial (Kipf and Welling, 2017).

In this model, a straight-forward (i.e., no approximation) version of GCN is employed. The method can be considered as a linear combination of the weights of the $1,2,\ldots,k$-nearest neighbors of each node. In this way, the effects of the neighbors can be dynamically adjusted through the coefficients. The weights of the kth nearest neighbors can be computed through k times self-multiplications of the adjacency matrix. The 1 to kth adjacent matrices $A,A^2\oplus\ldots\oplus A^k$ are stacked together to form a $N\times(k*N)$ matrix. Then, a row-wise normalization is performed on the stacked matrix, as the spatial relations of each node are considered independently. The graph convolution feature $GC_t\in\mathbb{R}^{N\times N}$ is computed as:

$$GC_t=\left(W_{gc}\odot norm\left(\left[A\oplus A^2\oplus\ldots\oplus A^k\right]\right)X_t \tag{11}$$

where $W_{gc}\in\mathbb{R}^{N\times(k*N)}$ is a trainable weight matrix of the elements in the adjacency matrices. $\oplus$ denotes a vertical concatenation of matrices. $X_t\in\mathbb{R}^N$ is the input data at timestamp t, and *norm* is the row-wise normalization method performed on $A\oplus A^2\oplus\ldots\oplus A^k$.

The output graph convolution features of each timestamp are then fed to an LSTM layer. For the recurrent unit of timestamp t, the input gate i_t, the output gate o_t, the forget fate f_t, and the memory cell state $\widetilde{C}_t$ are computed as:

$$i_t=\sigma(W_i[H_{t-1}\oplus GC_t]+b_i) \tag{12}$$

$$o_t=\sigma(W_o[H_{t-1}\oplus GC_t]+b_o) \tag{13}$$

$$f_t=\sigma(W_f[H_{t-1}\oplus GC_t]+b_f) \tag{14}$$

$$\widetilde{C}_t=tanh(W_c[H_{t-1}\oplus GC_t]+b_c) \tag{15}$$

where σ is the sigmoid activation function and *tanh* is the tangent activation function. W_i, W_o, W_f, and W_c are weights of the matrices and b_i, b_o, b_f, adn b_c are bias. H_{t-1} is the hidden state of the last timestamp. Both of H_{t-1} and GC_t are of $\mathbb{R}^{N\times N}$ features.

As the weights of relations among nodes vary at each timestamp, the cell state of the last timestamp should be re-weighted before being merged with the current cell state. A cell state gate is added to the original LSTM model as:

$$C_t^*=\left(W_N\odot norm\left(\left[A\oplus A^2\oplus\ldots\oplus A^k\right]\right)C_{t-1} \tag{16}$$

where W_N is the matrix to re-weight the effects of each node's neighbors. The final cell state and hidden state of timestamp t is computed as:

$$C_t=f_t\odot C_{t-1}^*+i_t\odot\widetilde{C}_t \tag{17}$$

$$H_t=o_t\odot tanh(C_t) \tag{18}$$

4.3.2. TCLSTM

During preprocessing, using L sliding windows leads to an input size of L times the original input, and thus L times the number of parameters in a linear model. In this case, a dilation convolution module is applied to mix the original data with the seasonal features and reshape the input size back to N.

The dilation convolution module contains four time-convolution layers. A convolution layer of dilation of N and a kernel size of L is first applied to merge the original data with seasonal features. Then, the other three layers merge the information of adjacent timestamps. The size of the final output matrix is $\tau \times N$, where τ is the number of timestamps in the first-level sliding window.

The output of the time convolution module is then sent to an LSTM layer. The implementation of the LSTM is the same as the LSTM in Section 4.3.1.

4.3.3. Kalman filter

In this model, the data are not treated as ground truth but as measurements with noise. Therefore, the temporal relations and spatial relations between data values might slightly diverge from the truth. The Kalman filter is employed to denoise predicted values by minimizing the differences between the statistical distributions of the temporal relation model's predictions and the spatial relation model's predictions.

The spatial analysis module and the time series analysis module are merged with the Kalman filter. To avoid the time-consuming matrix multiplications in the traditional Kalman filter, a simplified version of the Kalman filter is employed. The basic idea is to scale the output of TCLSTM and GCLSTM at each timestamp with the variance of the outputs. Based on Eq. 9, the formula is:

$$H_{kf,t} = \frac{c * Var_{tc,t} * H_{gc,t} + (1-c) * Var_{gc,t} * H_{tc,t}}{Var_{tc,t} + Var_{gc,t} + c} \tag{19}$$

Where w is a trained parameter in the model. $H_{kf,t}$ is the merged hidden state. $Var_{gc,t}$ and $Var_{tc,t}$ are the variances of the hidden states of the TCLSTM module and the GCLSTM module, respectively. $H_{gc,t}$ and $H_{tc,t}$ are hidden states of the GCLSTM module and the TCLSTM module. The hidden states of GCLSTM and TCLSTM are then updated as:

$$H'_{gc,t} = \left(H_{gc,t} - H_{kf,t}\right) \oplus H_{kf,t} \tag{20}$$

$$H'_{tc,t} = \left(H_{tc,t} - H_{kf,t}\right) \oplus H_{kf,t} \tag{21}$$

Finally, The output of the spatiotemporal module is merged with the daily historical data encoder module by a fully connected layer.

4.3.4. Encoder-decoder

The encoder-decoder module is simply a stack of linear layers. The idea is to first extract high-level features of the predicted features by reducing the number of dimensions and then expanding the number of dimensions to the number of features plus the number of seasonal features.

4.3.5. Loss

MSE loss is employed to evaluate the model's ability to predict both the original features and the corresponding seasonal features. The feature to be predicted is X_{t+1} and the seasonal feature is S_{t+1}, the loss function can be written as:

$$MSELoss = \frac{1}{l * N} \times \left[\sum_{i=1}^{N}\left(y_i^O - \hat{y}_i^O\right)^2 + \sum_{i=1}^{N}\left(y_i^{S_1} - \hat{y}_i^{S_1}\right)^2 + \ldots + \sum_{i=1}^{N}\left(y_i^{S_L} - \hat{y}_i^{S_L}\right)^2\right] \tag{22}$$

Where L is the number of seasonal features; y^O is the ground truth of the original data; $\hat{y}^O$ is the predicted values of the original data; y^{S_l} is the ground truth seasonal features extracted by sliding window l; $\hat{y}^{S_l}$ is the predicted seasonal features of sliding window l. It is noted that both the y^O and the y^{S_l} represent the traffic data itself, but not the label of the data. So, our S-DKFN is an unsupervised model.

4.4. S-DKFN Algorithm

The algorithm of the prediction model is described in Algorithm 1. The S-DKFN model needs three inputs: the adjacency matrix and the original time sequence data for GCLSTM mudule, the original data and the seasonal features extracted during preprocessing for TCLSTM part, and hyperparameters of the time window of length τ and the number of hops of neighbors K.

As described on line 7, GCN leverages a GCN block to extract the spatial-dependency features of the input data. TCN fuses the original data and the seasonal features with a dilation TCN block. The outputs of these two modules are fed to LSTMs layer to be processed separately (line 10–16). The outputs of the LSTM layers are merged with the Kalman filter (line 19–21) and fed to the next timestamp. At the final timestamp, the outputs of LSTMs are merged with the Kalman filter and then sent to a simple encoder-decoder module to generate a prediction of both the original data and the corresponding seasonal features (line 23–26).

Algorithm 1 Workflow of the S-DKFN model.

```
  Input: A graph $G = (V, E, A)$ with features of the last $T_p$
  timestamps $\left\{X_{t-T_p+1}, X_{t-T_p+2}, \ldots, X_t\right\}$ and corresponding
  seasonal features $\left\{S_{t-T_p+1}, S_{t-T_p+2}, \ldots, S_t\right\}$ in a time window
  of length $\tau$. Number of hops of neighbors $K$.
  Output: A graph $G = (V, E, A)$ with features
  $\left\{X_{t+1}, X_{t+2}, \ldots, X_{t+T_{pred}}\right\}$ at timestamps
  $t+1, t+2, \ldots t+T_{pred}$.
1: $\widetilde{A} = norm\left(\left[A \oplus A^2 \oplus \ldots \oplus A^k\right]\right)$
2: $H_{gc,-1} \leftarrow [0]_{N\times F}$, $H_{tc,-1} \leftarrow [0]_{N\times F}$
3: $C_{gc,-1} \leftarrow [0]_{N\times F}$, $C_{tc,-1} \leftarrow [0]_{N\times F}$
4:
5: for $j$ in $\{t-T_p+1, t-T_p+2, \ldots, t\}$ do
6: // Use GCN and TCN to map the inputs to the same space as
   the hidden states of the last timestamp
7:     $GC_j \leftarrow W_{gc} \odot \widetilde{A}X_j$ , $TC_j \leftarrow TCN(X_j \oplus S_j)$
8:
9: // Process spatial and seasonal features with different LSTM
   modules
10:     $i_{gc,j} \leftarrow \sigma(W_{gc,i}[H_{gc,j-1} \oplus GC_j] + b_{gc,i})$,
   $i_{tc,j} \leftarrow \sigma(W_{tc,i}[H_{tc,j-1} \oplus TC_j] + b_{tc,i})$
11:     $o_{gc,j} \leftarrow \sigma(W_{gc,o}[H_{gc,j-1} \oplus GC_j] + b_{gc,o})$,
   $o_{tc,j} \leftarrow \sigma(W_{tc,o}[H_{tc,j-1} \oplus TC_j] + b_{tc,o})$
12:     $f_{gc,j} \leftarrow \sigma(W_{gc,f}[H_{gc,j-1} \oplus GC_j] + b_{gc,f})$,
   $f_{tc,j} \leftarrow \sigma(W_{tc,f}[H_{tc,j-1} \oplus GC_j] + b_{tc,f})$
13:     $\widetilde{C}_{gc,j} \leftarrow tanh(W_{gc,c}[H_{gc,j-1} \oplus GC_j] + b_{gc,c})$,
   $\widetilde{C}_{tc,j} \leftarrow tanh(W_{tc,c}[H_{tc,j-1} \oplus GC_j] + b_{tc,c})$
14:     $C^*_{gc,j} \leftarrow W_{gc,N} \odot \widetilde{A}C_{gc,j-1}$, $C^*_{tc,j} \leftarrow W_{tc,N} \odot \widetilde{A}C_{tc,j-1}$
15:     $C_{gc,j} \leftarrow f_{gc,j} \odot C^*_{gc,j-1} + i_{gc,j} \odot \widetilde{C}_{gc,j}$,
   $C_{tc,j} \leftarrow f_{tc,j} \odot C^*_{tc,j-1} + i_{tc,j} \odot \widetilde{C}_{tc,j}$
```

(continued on next page)

```
    $H_{gc,j} \leftarrow o_{gc,j} \odot tanh(C_{gc,j}), H_{tc,j} \leftarrow o_{tc,j} \odot tanh(C_{tc,j})$

// Merge the spatial hidden states and the seasonal hidden states with Kalman filter
    $H_{kf,j} = \frac{c* \ Var_{tc,j} \ * \ H_{gc,j}+(1-c)* \ Var_{gc,j} \ * \ H_{tc,j}}{Var_{tc,j}+Var_{gc,j}+c}$
    $H'_{gc,j} = (H_{gc,j} - H_{kf,j}) \oplus H_{kf,j}$
    $H'_{tc,j} = (H_{tc,j} - H_{kf,j}) \oplus H_{kf,j}$
$H_{t:t+T_{pred}} \leftarrow W_0 H_{kf,t} + b_0$ // Expend 1 timestamp to $T_{pred}$ timestamps
$H_{t:t+T_{pred}} \leftarrow H_{t:t+T_{pred}} \oplus S_{t:t+T_{pred}}$ // Combine prediction results with $S_{t+1}, \ldots, S_{t+T_{pred}}$
$H_{t:t+T_{pred}} \leftarrow W_3(W_2(W_1 H_{kf,t} + b_1) + b_2) + b_3$ // Encoder-decoder for noise filtering
**return** $H_{t:t+T_{pred}}$
```

4.5. Anomaly score and complexity analysis

By applying S-DKFN to a series of traffic network data, the prediction results from timestamp T_{first} to T_{last} are generated. T_{first} is the first timestamp that can be predicted with enough historical data, and T_{last} is the last timestamp to be predicted. For each timestamp $t \in [T_{first}, T_{last}]$ and each node v, the anomaly score is computed based on the MSE formula:

$$Score = mean\left((y_{t,v} - \hat{y}_{t,v})^2\right) \tag{23}$$

Note that the $mean()$ function is applied over all the features if more than 1.

Based on the survey of Wu et al. (2020), the time complexity of a GCN layer is $O\left(md + nd^2\right)$ where m is the number of edges; n is the number of nodes; d is the number of features. Denote the sequence length as l and the number of seasons as s. It can be easily computed that the time complexity of TCN is $O(ln)$ and those of Linear layers are $O(n^2)$. In this case, the overall time complexity is $O\left(sl\left(md + nd^2 + ln\right) + n^2\right)$. As s and l are usually small, the time complexity can be simplified as $O\left(md + nd^2 + n + n^2\right)$. Similarly, the space complexity is $O\left(nd + d^2 + n^2\right)$.

5. Experiment

In this section, we explore how the duration and coverage of anomalies affect the efficiency of anomaly detection. The duration here indicates how many consecutive timestamps the anomaly spans, and coverage refers to how many adjacent nodes the anomaly covers. The experiments were designed based on the assumption that extensive duration and coverage of anomalies reduces the normal information the model acquires during prediction. Due to the lack of normal information, the predicted values may be closer to the abnormal and local features than to the normal and global patterns. The performance of our proposed model was compared with different types of state-of-the-art models. Besides, ablation experiments were also conducted to prove the effectiveness of each module of S-DKFN.

5.1. Dataset

We conducted vast experiments on two real-world datasets, including METR-LA and PEMS08 to evaluate the performance of S-DKFN. Particularly, METR-LA was used to examine the model performance against different duration. PEMS08 was used to simulate the duration and coverage of real accidents. We followed a similar process as in existing anomaly detection studies to inject artificial anomalies (Riani et al., 2009). Note that there are implicit anomalies in the datasets. The anomaly example of this sort is described in Introduction section.

5.1.1. METR-LA

METR-LA is a traffic network dataset containing 207 detectors. We extracted the speed measurements from May 1, 2012, to June 1, 2012, aggregated to 5-min average speeds. The missing rate was 2.6%. The missing data were filled using linear interpolation.

Adjacency Matrix. The adjacency matrix was generated based on the assumption that the closer a pair of nodes, the more likely they are to be connected. The geodesic distances on Earth between nodes were computed and normalized. Distances smaller than a threshold were chosen to be edges of the network. The value of the threshold was chosen through an experiment designed in Li et al. (2017).

Anomaly Generation. METR-LA were used to evaluate how different models deal with different durations of anomalies. For every 400 timestamps, an anomaly 5–10 mph smaller/larger than the minimum/maximum measurements per detector was added to the original dataset. The duration of the anomalies varies from 10 (50 min) to 50 (250 min) with a step size of 10.

5.1.2. PEMS08

The PEMS08 dataset was formed by the 5-min average speeds from September 1, 2019 to October 1, 2019. The data were collected by 1150 detectors along several arterial roads in Los Angeles. The missing rate was 0.14%, and the missing data were filled using linear interpolation.

Adjacency Matrix. The adjacency matrix was generated through the same strategy as for METR-LA. The average degree was 7.4. The mean and median of the length of edges were 262.5 m and 177.2 m, respectively. The minimum length were 0.0 m (self-loop), and the maximum length was 799.2 m.

Anomaly Generation. The PEMS08 dataset was used to simulate the spatial and temporal distribution of real traffic accidents. The measurements of detectors were within a 300 m/1000 m/3000 m and the duration of an accident was assigned to values 5–10 mph less than the minimum of the corresponding detectors' measurements.

5.2. Metrics

Anomaly detection problems are solved by distinguishing anomalies from normality. Therefore, anomaly detection problems are usually identified as binary classification problems. Precision, recall, F1-score, and AUC-ROC are efficient metrics to examine if a model can separate relevant instances from irrelevant instances and are thus be employed for result evaluation in this research.

Among the four metrics, precision is the fraction of normal samples among the instances identified as "normal." Recall is the fraction of retrieved normal samples among all normal samples. F1-score can be acquired from precision and recall.

AUC-ROC is the area under the receiver operating characteristic (ROC) curve. ROC first ranks data samples by their scores (the higher the score, the more likely the sample is an anomaly) and then plots the accumulated true positive rate (TPR) against the false positive rate (FPR) with a variety of thresholds.

In the following section, "PRC" represents precision. "RCL" means recall. "F1" represents F1-score, and "AUC" means AUC-ROC.

5.3. Baselines

In order to verify the effectiveness of S-DKFN, we evaluate it against six baseline methods: HA, SVR, Prophet, Telemamnom, DKFN, ASTGCN, STAWnet. Particularly, HA, SVR, and Prophet are models based on statistical or regression models. Telemamnom is a time-series anomaly detection model, and DKFN, ASTGCN, and STAWnet are spatiotemporal forecasting models.

HA (Liu and Guan, 2004). Historical average (HA) considers the average value of the last several timestamps as the prediction value.

SVR (Smola and Schölkopf, 2004). Support vector regression (SVR) is a piecewise linear regression model that minimizes the residuals of points very far from the line.

Facebook Prophet (F.C.D.S. team, 2017). Prophet is a piecewise regression model leveraging historical, seasonal, and holiday records for prediction. The model is claimed to be able to be used for anomaly detection in Medico (2020).

Telemanom (Hundman et al., 2018). Telemanom is a prediction-based anomaly detection model using LSTM. A batch size of 64 and a learning rate of 0.0001 are employed.

DKFN (Chen et al., 2020). DKFN is a state-of-the-art deep spatiotemporal forecasting model. The model also leverages GCN, LSTM, and the Kalman filter. The author claims the model is noise resistant. A batch size of 64 and a learning rate of 0.0001 are employed.

ASTGCN (Guo et al., 2019). ASTGCN is a state-of-the-art deep spatiotemporal forecasting model using an attention mechanism, GCN, and TCN. The model also leverages seasonal features to produce robust prediction results. A batch size of 64 and a learning rate of 0.001 are employed.

STAWnet (Tian and Chan, 2021). STAWnet (Spatial–Temporal Attention Wavenet) is one of the newest spatiotemporal forecasting models using an attention mechanism, GCN, and TCN. The model utilizes hierarchical temporal features and attention to improve performance. Our experiments employed the same hyper-parameters as in the paper.

5.4. Parameter settings

In the experiment, we used one feature (traffic speed) and one seasonal feature (daily) to train the model. The parameters for seasonal feature extraction can be determined by investigating the time decomposition results with different periods. The last 10 timestamps and their 7-day seasonal features were employed for prediction. The k in "k-hop neighbors" for GCN was set to be 3. In the TCN layers, all the kernel sizes along the time axis were 3. During training, the batch size was 64. An Adam optimizer with a learning rate of $1e^{-4}$ was employed.

For anomaly detection, a threshold over MSE is employed to separate anomalies from normal data. MSE was chosen over mean average error (MAE) and root mean average error (RMSE) due to its ability to exaggerate the differences between anomalies and normal data. Based on our experiment, a threshold of 800–1000 works well for all the scenarios. We chose 800 in the experiments below.

5.5. Anomaly duration analysis

The PRC, RCL, and F1 of the different models are listed in Table 1. The AUC-ROC scores of the different models are listed in Table 2. In Table 1 and 2, $dur = d$ indicates the duration of each anomaly. The actual duration is $d * 5$ minutes.

Table 1 shows that our model outperformed the baseline models on datasets with different anomaly durations. The performances of TCLSTM and GCLSTM drop due to the lack of context information. The performances of S-DKFN and all baseline models increase as the length of the anomaly increases to 40. The cause of this might be that more and more anomalies overlay with regular traffic jams, and thus, identifying the normal pattern becomes easier. However, the performances of S-DKFN and all baseline models start to drop at $duration = 50$, except for that of Prophet. The reason for the performance drop is revealed in Fig. 4, which indicates that the anomalies show a seasonal pattern. S-DKFN utilizes the previous seven days' seasonal features, so the performances drop if most of the seven days are covered with anomalies. However, Prophet leverage all the historical data to compute its seasonal features and is, therefore, less sensitive to periodical anomalies. Overall, Prophet and S-DKFN can usually outperform models without seasonal features in long-duration anomaly detection. ASTGCN, however, also uses seasonal features but is not able to produce competitive results for this dataset. S-DKFN always pose the highest AUC-ROC score for anomaly durations from 10 to 50 according to Table 2, except that the AUC-ROC score of S-DKFN (NE) is slightly higher than that of S-DKFN when $dur = 20$.

5.6. Anomaly coverage analysis

In this experiment, we explored whether the number of abnormal neighbors affect the performance of the models. As the maximum edge length was approximately 800 m, 3000 m was chosen to make sure that none of the reachable neighbors were normal.

Considering that the anomaly detection methods for time series (HA, SVR, Telemanom, and Prophet) cannot leverage the spatial relations information, the time series anomaly detection methods are only examined on the 300 m–anomaly dataset. The precision, recall, F1-score, and AUC-ROC are shown in Table 3.

Table 4 presents the results of the datasets with various anomaly coverage. Based on the results, a radius of 1000 m can negatively affect DKFN and S-DKFN, but each of the models performs worse with an anomaly coverage of 3000 m. Besides, STAWnet is not affected by radius of anomalies.

5.7. Case study

To examine how the prediction models handle the anomalies, a case study was performed on a detector (ID: 801246) from the 7000th time slot (September 25, 2019, 08:20:00, local time) to the 7600th time slot (September 27, 2019, 09:20:00, local time). Fig. 5 shows how speed varies as incidents happen. Our target detector is affected by five incidents during the 3000-min period, with two incidents overlapping with each other. The prediction results are plotted in Fig. 6. The plots are divided into four groups

Table 1
Precision, recall, and F1-score on METR-LA with different anomaly durations.

	METR-LA (dur = 10)			METR-LA (dur = 20)			METR-LA (dur = 40)			METR-LA (dur = 50)		
	PRC	RCL	F1	PRC	RCL	F1	PRC	RCL	F1	PRC	RCL	F1
HA	0.034	0.072	0.047	0.063	0.088	0.074	0.154	0.135	0.144	0.205	0.150	0.174
SVR	0.086	0.168	0.114	0.150	0.160	0.155	0.424	0.289	0.343	0.550	0.389	0.456
Prophet	0.450	0.496	0.472	0.682	0.587	0.631	0.809	0.864	0.836	0.756	**0.902**	**0.823**
Telemanom	0.253	0.416	0.315	0.440	0.504	0.470	0.506	0.338	0.406	0.579	0.316	0.409
ASTGCN	0.034	0.048	0.040	0.251	0.343	0.290	0.348	0.575	0.434	0.329	0.133	0.189
DKFN	0.418	**0.598**	0.492	0.485	0.495	0.490	0.364	0.433	0.396	0.386	0.418	0.402
STAWnet	0.281	0.063	0.103	0.323	0.063	0.106	0.385	0.041	0.074	0.413	0.034	0.062
TCLSTM[1]	0.519	0.589	0.552	0.327	0.419	0.367	0.443	0.605	0.511	0.494	0.435	0.462
GCLSTM[2]	**0.525**	0.584	**0.553**	0.324	0.418	0.365	0.438	0.635	0.518	0.492	0.433	0.460
S-DKFN (NE)[3]	0.326	0.438	0.374	0.755	**0.663**	0.706	0.443	0.263	0.330	0.384	0.346	0.364
S-DKFN	0.430	0.593	0.499	**0.819**	0.638	**0.717**	**0.876**	**0.906**	**0.891**	**0.831**	0.774	0.802

1. S-DKFN without GCLSTM and Kalman filter modules.
2. S-DKFN without TCLSTM and Kalman filter modules.
3. S-DKFN without the encoder-decoder module.

Table 2
AUC-ROC score on METR-LA with different anomaly duration.

	dur = 10	dur = 20	dur = 30	dur = 40	dur = 50
HA	0.520	0.510	0.507	0.656	0.504
SVR	0.736	0.801	0.845	0.865	0.872
Prophet	0.923	0.967	0.983	0.982	0.969
Telemanom	0.741	0.778	0.746	0.669	0.598
ASTGCN	0.544	0.680	0.748	0.794	0.645
DKFN	0.926	0.886	0.754	0.701	0.656
STAWnet	0.709	0.632	0.586	0.570	0.541
TCLSTM[1]	0.960	0.938	0.905	0.925	0.908
GCLSTM[2]	0.963	0.943	0.905	0.926	0.908
S-DKFN (NE)[3]	0.926	**0.985**	0.668	0.613	0.697
S-DKFN	**0.963**	0.977	**0.990**	**0.990**	**0.972**

1. S-DKFN without GCLSTM and Kalman filter modules.
2. S-DKFN without TCLSTM and Kalman filter modules.
3. S-DKFN without the encoder-decoder module.

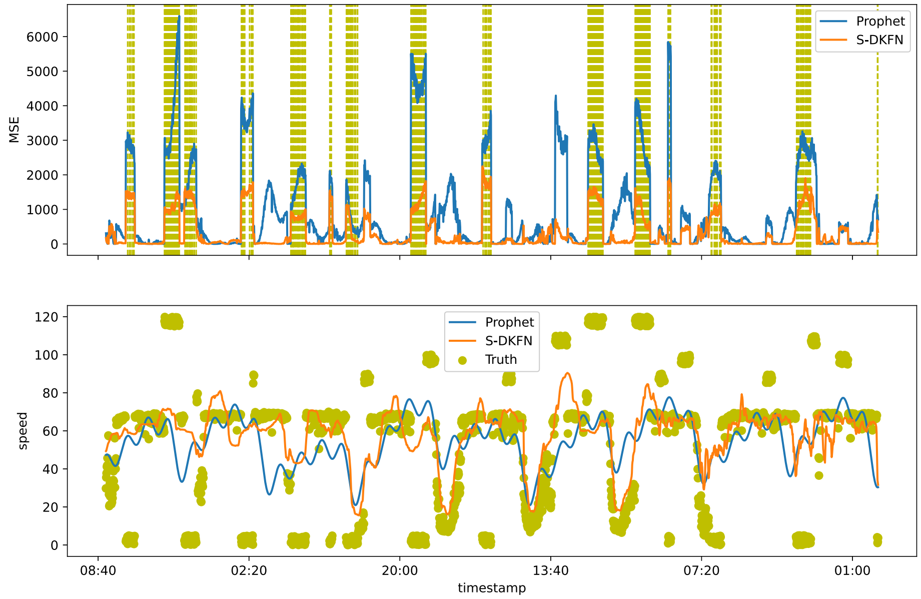


Fig. 4. MSE and prediction plot of Prophet and S-DKFN (dur = 50). Yellow dashed lines are timestamps with anomalies. Yellow dots are ground truth speeds. The x-axis is the index of timestamps.

Table 3
Anomaly detection result on PEMS08 (300 m).

	PRC	RCL	F1	AUC
HA	0.308	0.310	0.309	0.782
SVR	0.232	0.232	0.232	0.608
Prophet	0.720	0.747	0.733	0.987
Telemanom	0.222	0.241	0.231	0.801
ASTGCN	0.742	0.750	0.746	0.962
DKFN	0.554	0.567	0.561	0.962
STAWnet	0.472	0.030	0.057	0.693
TCLSTM[1]	0.620	0.804	0.700	0.992
GCLSTM[2]	0.646	0.796	0.713	0.992
S-DKFN (NE)[3]	0.315	**0.825**	0.456	0.986
S-DKFN	**0.745**	0.761	**0.753**	**0.993**

1. S-DKFN without GCLSTM and Kalman filter modules.
2. S-DKFN without TCLSTM and Kalman filter modules.
3. S-DKFN without the encoder-decoder module.

to ensure that all the lines are clearly shown. Each plot contains two parts. The first part displays the MSE against timestamps. Areas with yellow shadows indicate the time slots with anomalies. The second part shows the predicted speeds and the ground truth speeds against timestamps. The indices of the timestamps are used for the x-axis to better indicate the time interval among the ticks.

Based on Fig. 6, it can be concluded that methods without seasonal features (HA, SVR, Telemanom, DKFN, STAWnet) are usually not able to identify anomalies. Prophet performs well in anomaly detection and shows a tendency toward underfitting regular traffic jams (at around 08:20). S-DKFN (NE) appears to be a horizontal line, which means that the model without the encoder-decoder is not complex enough to simulate the time series. Both S-DKFN and ASTGCN are able to fit regular traffic jams while not being trapped by anomalies. However, none of the models can distinguish anomalies from seasonal traffic jams (at around 01:00). Both Fig. 4 and Fig. 6c show the robustness of the model. Taking the Fig. 6c as an example, our model can produce smooth predicted time series without being affected by noise. Since S-DKFN leverages different neural network architectures and denoising techniques, and the TCN module assembles different levels of seasonal features, which ensures that no noise by irrelevant timestamps is introduced. Furthermore, the encoder-decoder module expands single-season features to multi-seasonal features, forcing the model to drop some small-scale features to enhance the robustness of S-DKFN.

Table 4
Anomaly detection result on PEMS08 (1000 m and 3000 m).

	r = 1000				r = 3000			
	PRC	RCL	F1	ROC	PRC	RCL	F1	ROC
ASTGCN	0.393	0.382	0.387	0.737	0.409	0.273	0.327	0.849
DKFN	0.395	0.729	0.513	0.903	0.498	0.296	0.371	0.732
STAWnet	0.491	0.050	0.091	0.624	0.501	0.044	0.081	0.620
GCLSTM[2]	0.270	0.432	0.332	0.976	0.316	0.378	0.345	0.910
S-DKFN (NE)[3]	0.280	0.322	0.300	0.965	0.324	0.449	0.376	0.910
S-DKFN	**0.647**	**0.843**	**0.732**	**0.984**	**0.892**	**0.787**	**0.836**	**0.946**

1. S-DKFN without TCLSTM and Kalman filter modules.
2. S-DKFN without the encoder-decoder module.

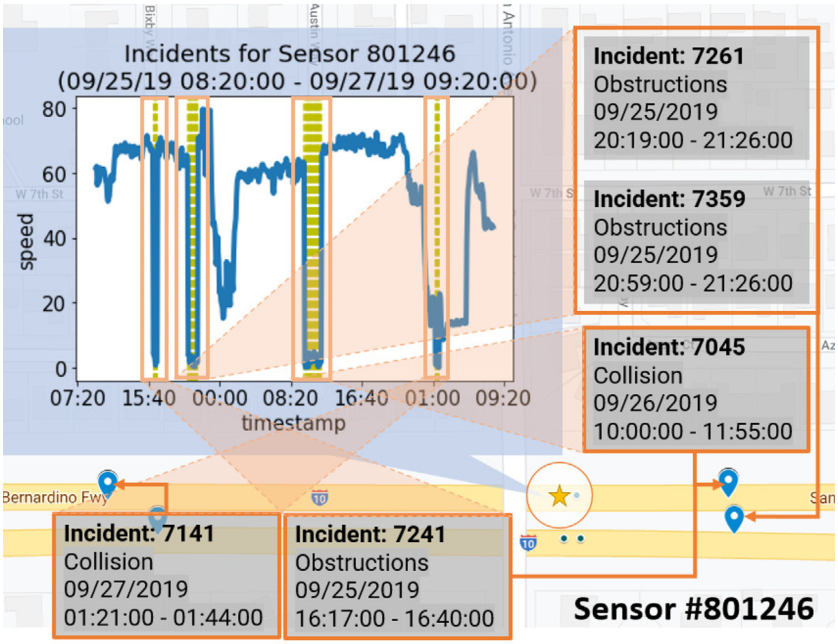
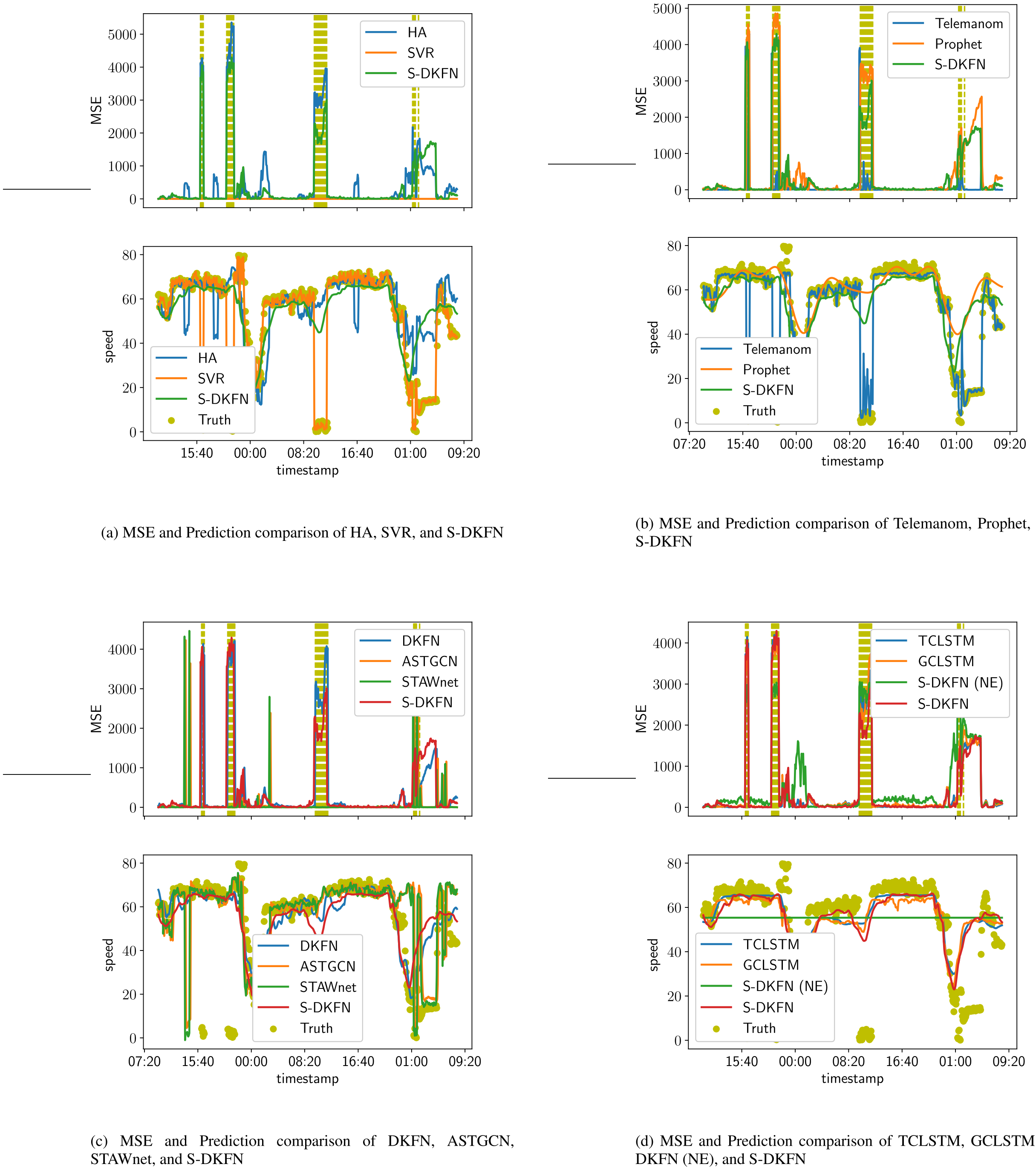


Fig. 5. Case study of detector #801246. The star marks the position of the #801246 and the blue markers are the locations of the accidents. The plot shows the detected speed variation (blue line) and the timestamps with incidents (yellow dashed lines).



(a) MSE and Prediction comparison of HA, SVR, and S-DKFN

(b) MSE and Prediction comparison of Telemanom, Prophet, and S-DKFN

(c) MSE and Prediction comparison of DKFN, ASTGCN, STAWnet, and S-DKFN

(d) MSE and Prediction comparison of TCLSTM, GCLSTM, S-DKFN (NE), and S-DKFN

Fig. 6. Case study of detector #801246. The upper part is the MSE plot of [HA, SVR, S-DKFN], [Telemanom, Prophet, S-DKFN], [DKFN, ASTGCN, STAWnet, S-DKFN], and [TCLSTM, GCLSTM, S-DKFN (NE), S-DKFN] against true anomalies (yellow dashed lines). The lower part is the plots of predicted speeds and ground truth (yellow dots).The x-axis is the time.

6. Conclusion

This paper presents S-DKFN, an unsupervised prediction-based anomaly detection model. S-DKFN incorporates various neural network architectures (GCN, TCN, LSTM, and encoder-decoder) and leverages time series decomposition and a Kalman filter for denoising. The model was examined on two traffic network datasets with artificial anomalies of different durations and coverages. The results show that S-DKFN can outperform all the baseline models in anomaly detection tasks. The case study indicates that S-DKFN can distinguish anomalies from regular traffic congestions. In addition, S-DKFN is robust to anomalies with broad coverages and long durations up to approximately four hours, and thus can be used for accident detection. In the future, we will use the correlation between traffic data of multi-types to form heterogeneous networks to learn more effective representations. Combining self-supervised contrastive learning, we will find more effective models to identify anomalous traffic behaviors.

Declaration of Competing Interest

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

Acknowledgements

The authors would like to thank the anonymous reviewers for their careful review and constructive comments. Thank Jason Tao from DDOT for providing the incident records. This work is supported by the VT Open Access Subvention Fund of Virginia Tech Libraries.

References

Ahmed, M.S., Cook, A.R., 1979. Analysis of freeway traffic time-series data by using Box-Jenkins techniques, no. 722.

Barbagli, B., Bencini, L., Magrini, I., Manes, G., Manes, A., 2011. A real-time traffic monitoring based on wireless sensor network technologies. In: Proceedings of 7th International Wireless Communications and Mobile Computing Conference, IWCMC '11, pp. 820–825.

Battaglia, P., Pascanu, R., Lai, M., Jimenez Rezende, D.,, kavukcuoglu, k., 2016. Interaction networks for learning about objects, relations and physics. In: D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, R. Garnett (Eds.), Proceedings of the 30th Advances in Neural Information Processing Systems, NIPS '16, Curran Associates Inc. pp. 1–9.

Bendre, S.M., 1989. Masking and swamping effects on tests for multiple outliers in normal sample. Commun. Stat.- Theory Methods 18 (2), 697–710.

Bruna, J., Zaremba, W., Szlam, A., Lecun, Y., 2014. Spectral networks and locally connected networks on graphs. In: Proceedings of 2nd International Conference on Learning Representations, ICLR '14, pp. 1–14.

Chen, F., Chen, Z., Biswas, S., Lei, S., Ramakrishnan, N., Lu, C.-T., 2020. Graph convolutional networks with kalman filtering for traffic prediction. In: Proceedings of the 28th International Conference on Advances in Geographic Information Systems, SIGSPATIAL '20, pp. 135–138.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., Bengio, Y., 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing, EMNLP '14. Association for Computational Linguistics, pp. 1724–1734.

Chow, A.H., Santacreu, A., Tsapakis, I., Tanasaranond, G., Cheng, T., 2014. Empirical assessment of urban traffic congestion. J Adv. Transp. 48 (8), 1000–1016.

Coskun, H., Achilles, F., DiPietro, R., Navab, N., Tombari, F., 2017. Long short-term memory kalman filters: Recurrent neural estimators for pose regularization. In: Proceedings of the IEEE International Conference on Computer Vision, ICCV '17. pp. 5524–5532.

Deb, R., Liew, A.W.-C., 2019. Noisy values detection and correction of traffic accident data. Inf. Sci. 476, 132–146.

Djenouri, Y., Belhadi, A., Lin, J.C.-W., Djenouri, D., Cano, A., 2019. A survey on urban traffic anomalies detection algorithms. IEEE Access 7, 12192–12205.

Doersch, C., 2021. Tutorial on variational autoencoders.

Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., Adams, R.P., 2015. Convolutional networks on graphs for learning molecular fingerprints. In: Proceedings of the 28th International Conference on Neural Information Processing Systems. MIT Press, Cambridge, MA, USA, pp. 2224–2232.

Enders, W. Applied econometric time series."2nd ed." New York (US): University of Alabama.

F.C.D.S. team, 2017. facebook prophet. URL: https://github.com/facebook/prophet..

Guo, S., Lin, Y., Feng, N., Song, C., Wan, H., 2019. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In: Proceedings of 33rd AAAI Conference on Artificial Intelligence, AAAI '19, pp. 922–929.

Hamed, M.M., Al-Masaeid, H.R., Said, Z.M.B., 1995. Short-term prediction of traffic volume in urban arterials. J. Transp. Eng. 121 (3), 249–254.

He, Z., Xu, X., Deng, S., 2003. Discovering cluster-based local outliers 24 (9–10), 1641–1650.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9 (8), 1735–1780.

Hundman, K., Constantinou, V., Laporte, C., Colwell, I., Soderstrom, T., 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding, arXiv preprint arXiv:1802.04431. pp. 387–395.

Kipf, T.N., Welling, M., 2017. Semi-supervised classification with graph convolutional networks. In: Proceedings of 5th International Conference on Learning Representations, ICLR '17. pp. 1–14.

Knorr, E.M., Ng, R.T., 1999. Finding intensional knowledge of distance-based outliers. In: Proceedings of the 25th International Conference on Very Large Data Bases, VLDB '99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA. pp. 211–222.

Li, Y., Yu, R., Shahabi, C., Liu, Y., 2017. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. In: Proceedings of 5th International Conference on Learning Representations, ICLR '18, pp. 1–16.

Liu, J., Guan, W., 2004. A summary of traffic flow forecasting methods. J. Highway Transp. Res. Develop. 3, 82–85.

Liu, W., Zheng, Y., Chawla, S., Yuan, J., Xing, X., 2011. Discovering spatio-temporal causal interactions in traffic data streams. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '11. Association for Computing Machinery, New York, NY, USA, pp. 1010–1018.

Liu, W., Luo, W., Lian, D., Gao, S., 2018. Future frame prediction for anomaly detection – A new baseline. In: Proceedings of 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR '18. IEEE Computer Society, pp. 6536–6545.

Liu, F., Xue, S., Wu, J., Zhou, C., Hu, W., Paris, C., Nepal, S., Yang, J., Yu, P.S., 2020. Deep learning for community detection: Progress, challenges and opportunities. In: Proceedings of 29th International Joint Conference on Artificial Intelligence, IJCAI '20. pp. 1–7.

Lu, W., Cheng, Y., Xiao, C., Chang, S., Huang, S., Liang, B., Huang, T., 2017. Unsupervised sequential outlier detection with deep architectures. IEEE Trans. Image Process. 26 (9), 4321–4330.

Lu, G., Ouyang, W., Xu, D., Zhang, X., Gao, Z., Sun, M.-T., 2018. Deep kalman filtering network for video compression artifact reduction. In: Proceedings of the European Conference on Computer Vision, ECCV '18, pp. 568–584.

Ma, X., Wu, J., Xue, S., Yang, J., Zhou, C., Sheng, Q.Z., Xiong, H., Akoglu, L., 2021. A comprehensive survey on graph anomaly detection with deep learning, pp. 1–26. arXiv eprint arXiv:2106.07178.

Medico, R., 2020. awesome-ts-anomaly-detection. pp. 135–138.

Megalingam, R.K., Mohan, V., Leons, P., Shooja, R., A.M., 2011. Smart traffic controller using wireless sensor network for dynamic traffic routing and over speed detection. In: Proceedings of IEEE Global Humanitarian Technology Conference, GHTC '11. pp. 528–533.

Moya, M.M., Koch, M.W., Hostetler, L.D., 1993. One-class classifier networks for target recognition applications. NASA STI/Recon Technical Report N 93, 24043.

Pang, L.X., Chawla, S., Liu, W., Zheng, Y., 2013. On detection of emerging anomalous traffic patterns using gps data. Data Knowl. Eng., 357–373

Pang, G., Shen, C., Cao, L., Hengel, A.V.D., 2021. Deep learning for anomaly detection: A review. ACM Comput. Surveys 54 (2), 1–38.

Rabiner, L.R., 1990. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 267–296.

Ramaswamy, S., Rastogi, R., Shim, K., 2000. Efficient algorithms for mining outliers from large data sets. SIGMOD RECORD 29 (2), 427–438.

Riani, M., Atkinson, A.C., Cerioli, A., 2009. Finding an unknown number of multivariate outliers. J. R. Stats Soc. Ser. B 71 (2), 447–466.

Roth, V., 2005. Outlier detection with one-class kernel fisher discriminants. In: Proceedings of Advances in Neural Information Processing Systems, NIPS '05. pp. 1169–1176.

Rumelhart, D.E., Hinton, G.E., Williams, R.J., 1985. Learning internal representations by error propagation (Tech. rep.). California Univ San Diego La Jolla Inst for Cognitive Science.

Schlegl, T., Seeböck, P., Waldstein, S., Schmidt-Erfurth, U., Langs, G., 2017. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery, 146–157.

Smola, A., Schölkopf, B., 2004. A tutorial on support vector regression. Stat. Comput. 14 (3), 199–222.

Su, X., Xue, S., Liu, F., Wu, J., Yang, J., Zhou, C., Hu, W., Paris, C., Nepal, S., Jin, D., Sheng, Q.Z., Yu, P.S., 2021. A comprehensive survey on community detection with deep learning, pp. 1–31. arXiv eprint arXiv:2105.12584.

Tian, C., Chan, W.K.V., 2021. Spatial-temporal attention wavenet: A deep learning framework for traffic prediction considering spatial-temporal dependencies. IET Intel. Transport Syst. 15 (4), 549–561. https://doi.org/10.1049/itr2.12044.

Tisljaric, L., Fernandes, S., Caric, T., Gama, J., 2020. Spatiotemporal Traffic Anomaly Detection on Urban Road Network Using Tensor Decomposition Method, 674–688.

Valada, A., Burgard, W., 2017. Deep spatiotemporal models for robust proprioceptive terrain classification. Int. J. Robot. Res. 36 (13–14), 1521–1539.

van den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K., 2016. Wavenet: A generative model for raw audio. In: Proceedings of 5th International Conference on Learning Representations, ICLR '17. pp. 1–15.

Wu, Z., Pan, S., Long, G., Jiang, J., Zhang, C., 2019. Graph wavenet for deep spatial-temporal graph modeling. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI '19, International Joint Conferences on Artificial Intelligence Organization, pp. 1907–1913.

Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Philip, S.Y., 2020. A comprehensive survey on graph neural networks. IEEE Trans. Neural Networks Learn. Syst. 32 (1), 4–24.

Yang, J., Zheng, W.-S., Yang, Q., Chen, Y.-C., Tian, Q., 2020. Spatial-temporal graph convolutional network for video-based person re-identification. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR '20, pp. 3289–3299.

Ye, M., Peng, X., Gan, W., Wu, W., Qiao, Y., 2019. Anopcn: Video anomaly detection via deep predictive coding network. In: Proceedings of 27th ACM International

Conference on Multimedia. Association for Computing Machinery, New York, NY, USA, pp. 1805–1813.

Yu, B., Yin, H., Zhu, Z., 2018. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In: Proceedings of the 27th International Joint Conference on Artificial Intelligence, IJCAI '18, International Joint Conferences on Artificial Intelligence Organization, pp. 3634–3640.

Yu, L., Du, B., Hu, X., Sun, L., Han, L., Lv, W., 2021. Deep spatio-temporal graph convolutional network for traffic accident prediction. Neurocomputing 423, 135–147.

Zhu, L., Laptev, N., 2017. Deep and confident prediction for time series at uber. In: Proceedings of the IEEE International Conference on Data Mining Workshops, ICDMW '17. pp. 103–110.