TBSM: A Traffic Burst-Sensitive Model for Short-term Prediction under Special Events

Yilong Ren[a,b], Han Jiang[a], Nan Ji[c], Haiyang Yu [a,b*]

[a] School of Transportation Science and Engineering, Beihang University, Beijing, 100191, China

[b] Beihang Hangzhou Innovation Institute Yuhang, Hangzhou, 310023, China

[c] Intelligent Transport System (ITS) R & D Center, Shanghai Urban Construction Design and Research Institute (Group) Co., Ltd., Shanghai 200125, China

[*] Corresponding author. E-mail: hyyu@buaa.edu.cn.

Abstract:

Traffic prediction is an important management tool for traffic guidance and control and an effective decision-making tool to help travelers plan routes and avoid congested road sections. However, due to the transient and sudden nature of traffic bursts caused by events and data limitations, mainstream methods do not perform well in short-term traffic prediction for special events (SEs). To address this challenge, we propose a traffic burst-sensitive model (TBSM) for short-term traffic prediction. Specifically, we first define a new state unit with the short-term trend and observed state to represent both the burst case and usual case. Second, a state-and-trend unit similarity degree (SD) measurement method and increment-based prediction model are proposed. The key parameter of this model balances the weight of the short-term trend with the observed state. Finally, we use a deep deterministic policy gradient (DDPG) framework containing long short-term memory (LSTM) networks to realize the self-learning and adjustment of weights to ensure the generality and burst sensitivity of the model. The TBSM is implemented in the district of Beijing Workers' Stadium, where SEs occur frequently. The results demonstrate that the proposed model performs significantly better than other traditional machine learning approaches and deep learning approaches for SEs. Our TensorFlow implementation of the TBSM is available at https://github.com/buaajh/TBSM-Traffic-burst-sensitive-model-.

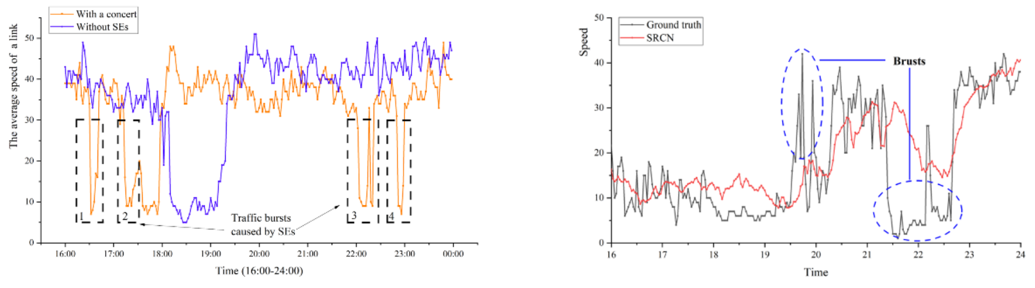
Key words: Short-term traffic prediction, special events, traffic burst prediction, deep reinforcement learning

1. Introduction

With the development of intelligent traffic systems, traffic prediction is receiving constant attention from researchers. Accurate traffic prediction can not only provide a scientific basis for traffic managers to sense traffic congestion and limit vehicles in advance but also help urban travelers choose appropriate travel routes and improve travel efficiency [1–3]. Benefiting from the massive amounts of available traffic data, data-driven methods, especially deep learning (DL) methods, have become the mainstream methods of traffic prediction. In recent years, a large number of DL models, including recurrent neural networks (RNNs) [4], convolutional neural networks (CNNs) [5], graph convolutional networks (GCNs) [6] and their variations and combinations, have been employed for traffic prediction and have achieved excellent results.

However, short-term traffic prediction under special events is still considered to be a serious challenge [7,8]. According to the National Highway Institute, U.S., a special event (SE) can be defined as an occurrence that abnormally increases traffic demand, including sporting events, concerts, traffic accidents and infrastructure failures [9]. These events are usually accompanied by ‘traffic bursts’ and cause a short time period of overload of road networks [10]. Combined with specific data, the difficulties of traffic prediction under the bursts caused by SEs are shown as follows:

- **Traffic bursts are hard to capture:** Since the time at which an SE occurs is unpredictable, forecasting models can be adjusted only after a burst is observed. As shown in Fig. 1(a), bursts caused by SEs generally occur many times and last for a short period. Thus, it would be difficult to capture traffic bursts without specifically characterizing them as new features, which leads to the accumulation of prediction errors.
- **Data sparsity problem of traffic bursts:** Data on traffic bursts are extremely scarce. According to [10] and [11], the proportion of burst data is less than 2% of the total data. Since the occurrence of bursts is rare, the prediction errors of the associated data points have a relatively minimal contribution to the total prediction errors, which makes it easy for prediction models to remember the patterns of usual data but not those of burst data.



(a) (b)

Figure 1. (a) shows the average speed of a link between 16:00 and 24:00 on July 11 (Saturday, with a concert) and July 18 (Saturday, without any planned SEs). (b) shows the visualization results for short-term prediction under SEs of the SRCN model.

Therefore, the performance of mainstream methods tends to be dissatisfactory in traffic states under special events. The core of conventional traffic flow prediction methods is to appropriately establish a learnable mapping relation that links historical records and future traffic flow data. This phenomenon can be explained from a probabilistic perspective: traffic bursts caused by SEs result in observations that are far from others [12]. In the context of a sequence prediction problem, for similar historical records, the conditional probabilities of subsequent values in unusual cases are relatively low, so that traffic bursts caused by SEs are discarded as outliers in most of the previous works. Taking an advanced DL model, the spatiotemporal recurrent convolutional network (SRCN), as an example, in short-term prediction, the model will disregard these outlier traffic bursts, as shown in Fig. 1(b). Similar results have been obtained in models such as the temporal graph convolutional network (T-GCN) [6] and reinforced spatial-temporal attention graph (RSTAG) [13].

Researchers have made some attempts at short-term traffic prediction under SEs. Some hope to enhance their forecasts with data from other sources, such as social media [14]; this idea is ingenious but unstable because it cannot handle emergencies. Some previous works try to label collected traffic flow data to distinguish usual cases and bursts [15]. They train separate models dedicated to different traffic states. However, this kind of method relies too much on manually calibrated data, and severely imbalanced training samples cannot support them. In addition, for highly dynamic traffic flow, especially unexpected traffic busts, prediction models must have highly dynamic adaptation ability. Most models do not yet have this ability.

Some studies show that traffic flow series collected from the same link over several consecutive days have similar trends; these trends are defined as intraday trends. In data-driven forecasting, traffic bursts are easy to disregard because they deviate too far from the intraday trend [10,16,17]. Inspired by this, we extract the short-term trends to characterize traffic bursts and fuse them with observed traffic states as a new feature to describe the evolution of traffic flow. Based on the abovementioned discussion, we propose a novel method referred to as the traffic burst-sensitive model (TBSM) for traffic forecasting tasks influenced by SEs. The TBSM applies state-and-trend units to conduct pattern recognition of traffic states and uses a new criterion of the similarity degree (SD) to mine 'high-value' data. In this way, the model can sensitively capture and characterize traffic bursts. Additionally, we use a deep deterministic policy gradient (DDPG) framework, which contains long short-term memory (LSTM) networks to realize the self-learning and adjustment of the TBSM to eliminate the dependence on massive annotated data [18]. Our contributions are threefold:

1. A novel model is constructed for short-term traffic prediction under SEs. By considering both the observed traffic state and its short-term evolution trend, the model realizes traffic burst sensitivity.
2. We formulate the traffic state prediction problem as a Markov decision process (MDP), in which the state space, action and reward function are skillfully designed. In addition, to maintain a real-time and self-adaptive model, we consider a continuous action space, and thus, a DDPG framework is proposed to solve the problem. To the best of our knowledge, similar modeling ideas and optimization methods have not been reported in traffic prediction problems.
3. We evaluate our approach using a real-world traffic dataset. The results show that this method has significant advantages in 2- to 6-min short-term traffic prediction. Further analysis demonstrates that the proposed model can better capture brief bursts of traffic flow.

The remainder of this paper is organized as follows. Section 2 reviews relevant research on traffic forecasting. In section 3, we describe the TBSM in detail. Section 4 provides some numerical results to verify the effectiveness of the proposed method.

Finally, we conclude the paper and discuss future work in section 5. The necessary proofs and algorithm details are available in the Appendix.

2. Literature review

Traffic flow prediction approaches can be divided into two major categories: model-based approaches and data-driven approaches [19]. Many model-based prediction approaches have emerged in previous years [20]. The representative methods include the traffic velocity model [21], cell transmission model [22], store and forward model [23], etc. Researchers have tried to explain the instantaneous and steady-state relationships among traffic volume, speed, and density through these models. However, the variations in traffic data in complex, real-world environments can rarely be described accurately, and the construction of these models is easily influenced by traffic disturbances and sampling point spacing. As a result, this kind of model has gradually fallen out of favor.

With the development of data acquisition technology, data-driven approaches have been widely considered with the support of massive traffic-related big data [24–26]. Compared with model-based approaches, such methods mainly infer the variation tendencies based on the statistical regularity of historical data and are universal and flexible. Over the past decade, there has been a proliferation of data-driven forecasting approaches; these methods can be divided into parametric models and nonparametric models [27,28]. Parametric models usually calibrate the parameters of the regression function through historical data to realize traffic state prediction. Due to the simplicity of the algorithms and convenient calculations, many parametric models have appeared. The autoregressive integrated moving average (ARIMA) model is the most representative. In 1995, Hamed et al. first used the ARIMA model to predict the traffic volume in urban arterials [29], and improvements based on this model have continuously emerged, including the Kohonen ARIMA [30], subset ARIMA [31] and seasonal ARIMA [32]. Although they are very easy to use, traditional parametric models can scarcely reflect the nonlinearities and uncertainties of traffic data, so their prediction accuracy is limited. The emergence of nonparametric models addresses these problems well. With enough historical data, the nonparametric models can learn statistical regularity and achieve higher accuracy. Common nonparametric models include K-nearest neighbor (KNN) models, support vector regression (SVR) models, and neural network models.

Influenced by the technological advancement of natural language processing and computer vision, DL prediction models based on artificial neural networks (ANNs) are favored by scholars for their learning and generalization abilities. For example, in [33], Huang et al. developed a multitask deep structure based on the unsupervised learning method of a deep belief network and with a supervised learning regression layer at the top to predict short-term traffic flow. Compared with the prediction accuracy of traditional methods such as ARIMA, support vector machines (SVMs) and ANNs, the prediction accuracy is improved by approximately 5%. Similar modeling methods also appear in [34] and [35]. With the development of research, people have gradually realized the importance of modeling spatiotemporal dependence. In this process, deep hybrid architectures consisting of CNNs and LSTM have gained favor [5]. Lv et al. proposed a stacked autoencoder (SAE) model to capture the spatiotemporal features from traffic data and to realize short-term traffic flow prediction [36]. Yu et al. proposed an SRCN model that combines a deep CNN and LSTM and achieved accurate short-term and long-term traffic state prediction [37], and Yang et al. developed a convolutional long-term memory neural network based on critical road sections to overcome the problem of structural missing data [38]. However, the spatial relationship of spatial monitoring points in road networks is non-Euclidean, which leads to the failure of CNN-based methods to reflect spatiotemporal dependence in essence. Therefore, the GCN is developed to model the spatial relationship. In [6], Zhao et al. used the GCN to obtain the spatial dependence and the gated recurrent unit (GRU) to obtain the temporal dependence and to construct a T-GCN model. The success of the T-GCN model has attracted extensive attention. Recently, researchers have focused on using graph networks to improve traffic prediction [39–41]. A series of models, such as the graph multiattention network (GMAN) [42], optimized graph convolution RNN (OGCRNN) [43] and geographically weighted gamma regression (GWGR) [44], were developed and obtained advanced effects.

Although short-term traffic state prediction has been a popular research topic for a long time, relatively few studies address prediction methods for traffic bursts caused by SEs. Some researchers tend to improve the quality of data. In [14], Ni et al. developed a short-term traffic flow model that incorporates features extracted from social media to forecast the incoming traffic flow prior to sport events. They extracted effective information from tweets to understand the attention and opinions of the public in relation to prior-event traffic prediction. However, since the judgment and classification of social media data are highly subjective, they are not persuasive enough. In addition, social data are prone to privacy and difficult to obtain. Therefore, the universality and real-time performance of this method are limited. In [16] and [45], detrending is employed in data processing to alleviate the influence of intraday trends on short-term prediction. This method removes long-term trends from the data input and improves the prediction accuracy to some extent, but it still cannot avoid the problem of ignoring transient traffic bursts. Other researchers try to use multiple models for both usual cases and special cases [8,15]. For example, in [15], the authors divide the data into usual case and accident data in advance, train two LSTM models and combine them. Such methods rely heavily on large amounts of preannotated data, and different event types degrade their performance. It can achieve good results only if we know which kind of SEs would take place and have enough sources to label the data. However, many types of SEs, such as traffic accidents and natural disasters, are unpredictable, leaving few options other than capturing them as they happen.

In general, current traffic flow prediction models for SEs still have some limitations; specifically, they tend to ignore the traffic bursts caused by events, lack real-time data and are overly reliant on labeled data. Different from previous studies, we establish a state-and-trend unit and directly search for similar features based on it. In this way, the model can sensitively capture traffic bursts. After Gaussian weighting of the parameters, we treat the prediction problem as a sequential decision and apply the reinforcement learning method to optimize it. Reasonable reward function settings eliminate the dependence on labeled data. Most importantly, this model is self-learning in real time based on traffic conditions, which is more practical.

3. Methodology

3.1 Problem statement

For the limitation of the existing models, we describe the concept and method introduced in the model and give the basic prediction form so that we can extract high-value data from sparse data and capture traffic bursts.

3.1.1 Definitions

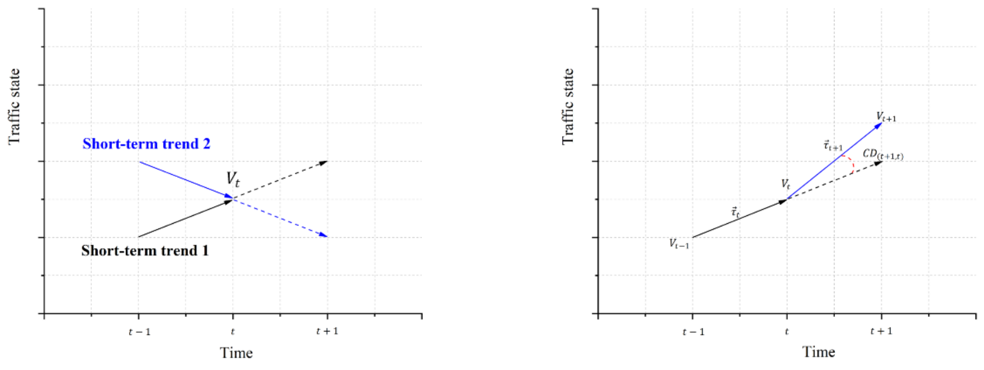
1. **Traffic state** V_t: The goal of this paper is to predict the traffic state in a certain period based on historical data. The traffic state is a general concept that can be the traffic speed, flow, or density. We denote the traffic state of link l_i at time t as $v_{l_i}(t)$. Furthermore, for road network $L = \{l_1, l_2, \dots, l_n\}$, its traffic state V_t can be defined as $V_t = \{v_{l_1}(t), v_{l_2}(t), \dots, v_{l_n}(t)\}$.
2. **Short-term trend** $\vec{\tau}_t$: As shown in Fig. 2(a), for the same traffic state V_t, different short-term trends may produce completely different evolutionary outcomes. Therefore, it is more advantageous to add further short-term trends than to use only state series for prediction under traffic bursts. Assume that δ is the time lag of prediction, and $\vec{\tau}_t$ is introduced to describe the short-term evolution of the traffic state. This evolution can be viewed as the trend direction of the traffic state evolving from one time $(t-\delta+1)$ to another time t in the n-dimensional space where the state point is located as follows:
$$\vec{\tau}_t = V_{t-\delta+1} - V_t \tag{1}$$
The short-term trend is a high-dimensional vector with the same dimension as the number of road sections in the road network, and we only take its direction as a feature.
3. **State-and-trend unit** X_t: To sensitively capture burst phenomena in traffic flow under the influence of SEs, we regard X_t as the attribute feature of the road network at time t. On the basis of V_t and $\vec{\tau}_t$, the state-and-trend unit can be described as follows:
$$X_t = \{V_t, \vec{\tau}_t\} \tag{2}$$

In this way, the traffic forecasting problem can be considered as learning the key parameters of prediction model P on the premise of historical data and the observed state-and-trend unit X_t and then calculating the traffic state V_{t+f} in the next f moments, as shown in (3).

$$[X_{t-\delta+1}, X_{t-\delta+2}, \dots, X_t] \xrightarrow{P} V_{t+f} \tag{3}$$

3.1.2 Similarity measurement of the state-and-trend units

For traffic bursts, nearest neighbor analysis (NNA), which can directly select the samples most similar to the current state from the state space instead of disregarding it, is a more appropriate prediction form [8,10]. As mentioned in section 1, forecasting can be regarded as a process of recognizing natural groups or clusters in multidimensional data based on similarity measures. Therefore, a novel measure of the SD between historical data and observed values should be constructed based on the state-and-trend unit. In this process, we choose to measure the traffic state and short-term trend that constitute the state-and-trend unit and then fuse them.



(a) Influence of short-term trends

(b) Employ the Cosine distance to measure the similarity between two short-term trends.

Figure 2. Graphic illustration of short-term trends. (a) describes the influence of short-term trends on the evolution of traffic states and (b) demonstrates the rationality of using cosine similarity to measure the similarity between short-term trends.

Traditional NNA-based models usually employ Euclidean distance (ED) to measure the similarities between the predicted state series and the archived state series [46,47]. In the proposed model, we followed this approach to address traffic states V_t. The ED between V_t and historical state V_{t_i} is calculated as follows:

$$\mathrm{ED}_i = \| V_t - V_{t_i} \|_2 \tag{4}$$

For short-term trends $\vec{\tau}_t$ and $\vec{\tau}_{t_i}$, we can regard them as two vectors in state space, starting from the same origin and pointing in different directions, and an angle is formed between them, as shown in Fig. 2(b). If the angle is 0 degrees, the direction is the same, which means that the short-term trends of t and $t+1$ are exactly equal, and vice versa. Therefore, we can judge the similarity of short-term trends by the size of the included angle. Cosine similarity measures the similarities between vectors by the cosine of their angle, which has been one of most practical similarity measures [48]. Nonpositive cosines usually exhibit opposite trends; thus, we define the cosine distance (CD) to reflect the similarity of traffic evolution trends, as shown in (5).

$$\mathrm{CD}_i = 1 - \frac{\vec{\tau}_t \cdot \vec{\tau}_{t_i}}{\|\vec{\tau}_t\| \|\vec{\tau}_{t_i}\|} \tag{5}$$

CD takes 1 minus cosine similarity; thus, it is bounded by $[0, 2]$. When CD is equal to 0, the angle between $\vec{\tau}_t$ and $\vec{\tau}_{t_i}$ is zero and these two short-term trends are regarded as identical. Meanwhile, when CD is 2, these two short-term trends are regarded as opposite.

The EDs between the current benchmark state and historical states are mapped to the scale of [0, 2], the same as CD, using the min-max scaling method. Thus, the SD between samples is:

$$\mathrm{SD}_i = \alpha \frac{2(\mathrm{ED}_i - \min\{\mathrm{ED}\})}{\max\{\mathrm{ED}\} - \min\{\mathrm{ED}\}} + (1-\alpha)\mathrm{CD}_i \tag{6}$$

where α is the factor to balance the weight of ED_i and CD_i in the forecast and $\alpha \in [0,1]$. We define it as the equilibrium factor. When α approaches 0, the short-term trend plays a decisive role in the measurement of state unit similarity. In contrast, the similarity of state units depends more on the similarity of reference state points when α approaches 1. Therefore, the dynamic property of α better adapts the similarity measurement for urban traffic states under SEs.

3.1.3 Foundational form of the prediction results

Assume that the searched K-nearest neighbors of X_t according to SD are $[X_{t_1}, X_{t_2}, \cdots, X_{t_K}]$ and the corresponding traffic state is $[V_{t_1}, V_{t_2}, \cdots, V_{t_K}]$. In the general form of NNA, the predicted value at $t+f$ can be expressed as (7).

$$\hat{y}_t = \frac{1}{K}\sum_{i=1}^{K} V_{t_i} \tag{7}$$

To reduce the interference of the difference caused by the randomness of different neighbors to the prediction, we choose to forecast the increments. We define the increment as the difference value between adjacent V_{t_i} and V_{t_i+f}; then, the form of the predicted results is shown as follows:

$$\begin{cases} \triangle y_{t_i} = V_{t_i+f} - V_{t_i} \\ \hat{y}_t = V_t + \frac{1}{K}\sum_{i=1}^{K} \triangle y_{t_i} \end{cases} \tag{8}$$

3.2 Problem transformation

3.2.1 Gaussian weighted prediction mode

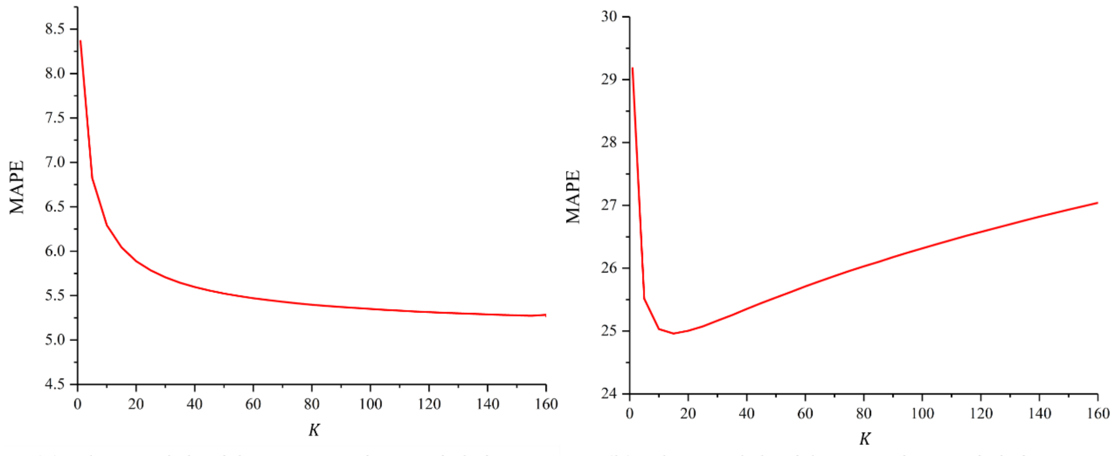
In section 3.1, we present the form of the increment-based traffic prediction. K, the number of nearest neighbors, is undoubtedly the basis of the model. The parameter α introduced in the new similarity determines whether the model can make predictions under traffic bursts. In [49], the authors point out that the Gaussian weighted prediction function integrates the generations of nearest neighbors so effectively that prediction accuracy will neither increase significantly nor decrease when K increases by more than a certain value, as shown in Fig. 3.

In other words, we can judiciously relax the constraint of K in the proposed model by Gaussian weighting and concentrate on choosing an appropriate α for the traffic state under SEs. The introduced Gaussian function is shown as follows:

$$w_i = e^{-2\mathrm{SD}_i{}^2} \tag{9}$$

and the relevant prediction model is transformed as (10).

$$\hat{y}_t = V_t + \sum_{i=1}^{K} \frac{w_i}{\sum_{i=1}^{K} w_i} \triangle y_{t_i} \tag{10}$$



(a) The model without Gaussian weighting. (b) The model with Gaussian weighting.

Figure 3. MAPE changes with the K value before and after Gaussian weighting.

3.2.2 MDP modeling of the prediction problem

In (10), the optimal value of K can be calibrated in advance through massive data, and we describe this process in Appendix A. However, for every observed state-and-trend unit X_t, there is a parameter α that minimizes the prediction error. A fixed prediction model obviously does not satisfy the need to address the traffic state under both SEs and the usual case, and a model that can constantly adjust itself is necessary. With that as the motivation, the prediction problem is transformed into a sequential decision problem; i.e., in each prediction window, we calculate an optimal α value in real time based on the observed state units and historical sequence to minimize the prediction error.

Using the absolute value of the prediction residual as the loss function, the problem can be expressed as follows:

$$\min \left| V_{t+f} - \left(V_t + \sum_{i=1}^{K} \frac{w_i}{\sum_{i=1}^{K} w_i} \left(V_{t_i+f} - V_{t_i} \right) \right) \right| \tag{11}$$

subject to:

$$\begin{cases} w_i = e^{-2SD_i^2} \\ \mathrm{SD}_i = \alpha \frac{2(\mathrm{ED}_i - \min\{\mathrm{ED}\})}{\max\{\mathrm{ED}\} - \min\{\mathrm{ED}\}} + (1-\alpha)\mathrm{CD}_i \\ \alpha \in [0,1] \end{cases}$$

where V_t is known. V_{t_i} and V_{t_i+f} are retrieved from the historical database according to SD, which only relates to α. However, V_{t+f} is completely unknown and can only be observed in the $t+f$ moment, which makes it unsolvable with conventional optimization methods. Considering that the short-term traffic prediction task usually adopts the form of a rolling forecast, that is, when a prediction is completed and the duration of the prediction window is extended, the real-time traffic condition obtained by the perception method can be used for the next prediction as well as the evaluation of the original prediction. Within each prediction step, the decision-making process depends only on the observed state $[X_{t-\delta+1}, X_{t-\delta+2}, \ldots, X_t]$, does not depend on the historical action of the model, and therefore conforms to the Markov property. In that case, the above sequential decision problem can be expressed as a continuous-time MDP. This modeling approach provides the possibility of using reinforcement learning (RL) to judge the state of the environment and to optimize the prediction model in real time.

3.3 DDPG-enabled TBSM

To adjust the model in real time and to effectively mitigate traffic bursts, we choose to design an agent to solve the MDP problem proposed in section 3.2. The agent aims to automatically provide the optimal prediction model configuration scheme according to the different short-term traffic state evolution scenarios and the prediction model's prediction effect evaluation in each prediction. Additionally, under the premise of fully learning the evolution characteristics of short-term traffic states, the model can fully evaluate the running state of the prediction model in real time, which is beneficial to real-time dynamic optimization of the prediction model and improves the prediction accuracy. In Fig. 4, we present an iteration of tasks in which an agent and an environment interact at discrete time steps.

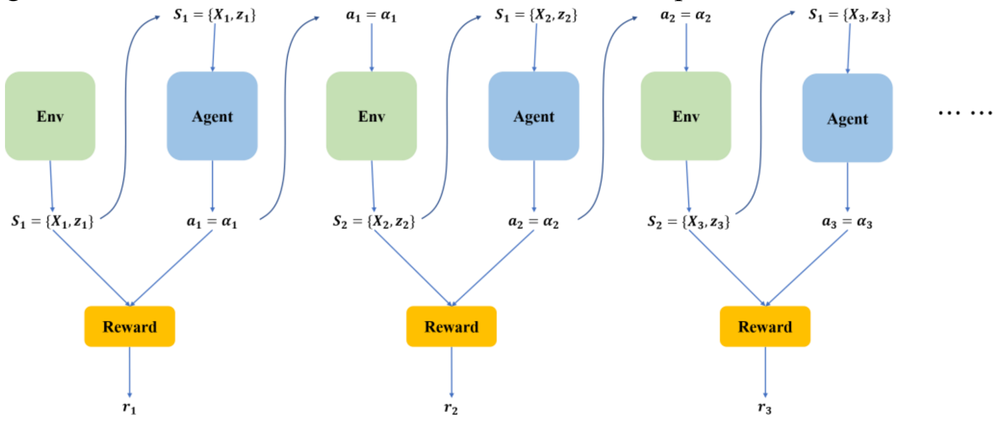


Figure 4. Iteration of prediction tasks.

To apply the above framework to traffic prediction, we first need to define some features to represent the condition, a set of actions and a reward function. Then, we should choose an appropriate algorithm to establish the optimal policy to maximize the rewards in each time step.

3.3.1 Design of deep reinforcement learning (DRL) components

In our model, three key elements of RL can be expressed as follows:

- **State** S_t: To ensure that the RL agent can learn a proper policy, it needs inputs that are representative of the traffic state and are somewhat predictive in aggregate. Therefore, we set $S_t = \{X_t, z_t\}$, where X_t is the state-and-trend unit at time t and z_t represents the state of the prediction model itself. We use the known residual of the last prediction at time t to describe this parameter, i.e., $z_t = V_t - \hat{y}_{t-f}$.
- **Set of actions** a_t: In the prediction problem presented in section 3.2, the equilibrium factor α in (6) is the key element. Therefore, a_t is defined as the process of choosing a value of α, where $\alpha \in [0,1]$.

- **Reward** r_t: The reward of an agent is defined as the average index improvement rate; that is, when the index obtained by executing a_t is smaller than that obtained by the precalibrated model P_0, a_t is considered valid; otherwise, the opposite is true. Thus, r_t can be calculated as follows:

$$r_t = \begin{cases} \frac{1}{2}\left(\frac{MAE_t - MAE_t'}{MAE_t} + \frac{MAPE_t - MAPE_t'}{MAPE_t}\right) \times 100\%, a_t \text{ is valid or invalid} \\ 0, a_t \text{ is incompletely valid} \end{cases} \tag{12}$$

where MAE_t and MAE_t' are the average absolute value errors obtained by P_0 and using action a_t, respectively. Similarly, $MAPE_t$ and $MAPE_t'$ can be obtained. If the indexes obtained by executing action a_t are all smaller than those obtained by P_0, we can regard action a_t as an effective optimization, and we define that $r_t > 0$. Otherwise, a_t is regarded as invalid optimization. To accelerate the convergence of the algorithm, we also define the actions with an optimization ratio less than 1% as 'incomplete valid'; the reward for the action is 0. Thus, the effect of prediction can be fed back to the agent with positive and negative real-time rewards so that the agent can learn how to select parameter α after repeated training to improve the prediction effect of the model. The acquisition process of P_0 is shown in Appendix A.

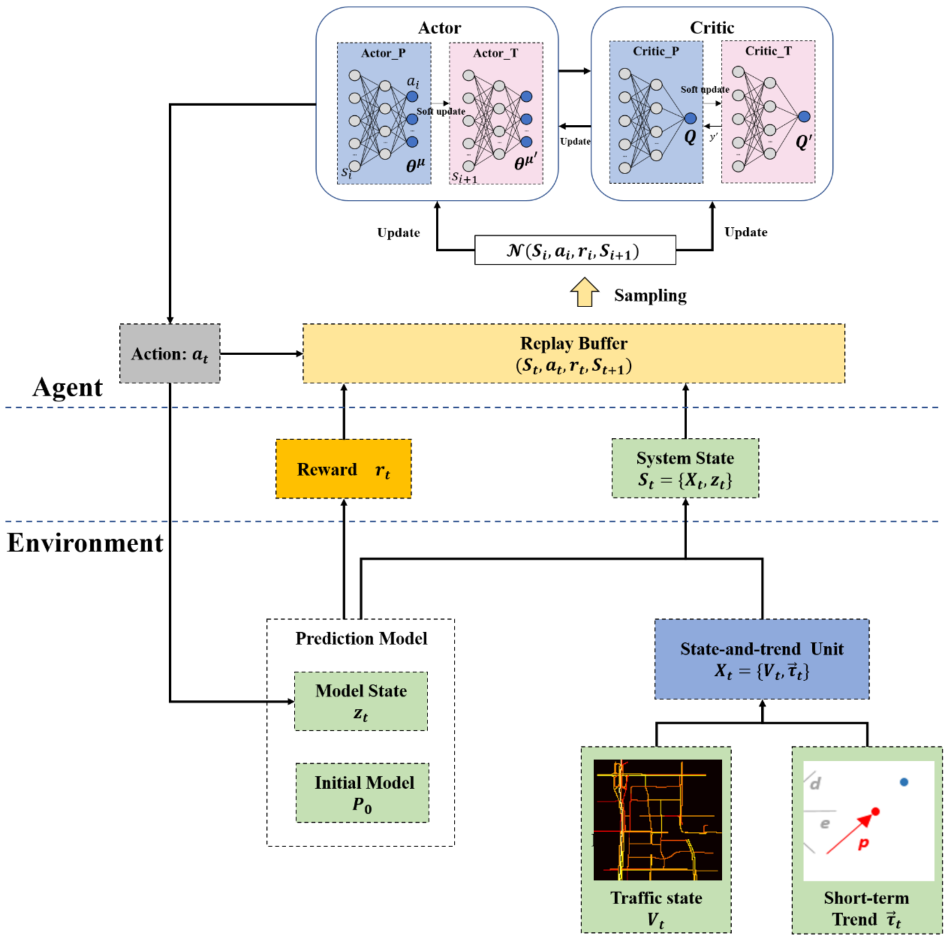


Figure 5. Framework of the proposed method.

3.3.2 Optimal strategy search algorithm: DDPG

Since the state is formulated by dynamic environmental information and the action space contains many continuous values, normal DRL methods such as a deep Q network (DQN) cannot handle this kind of problem. Suppose that DDPG is an actor-critic and model-free algorithm for RL over continuous action spaces and outputs deterministic actions in a stochastic environment to maximize cumulative rewards [50]. Therefore, we propose a DDPG-based method to maximize the reward function. The framework of the proposed method is shown in Fig. 5 and consists of an actor network, a critic network and a replay buffer. In addition, each network is constructed by two deep neural networks (DNNs), i.e., a primary network to select actions and a target network to evaluate actions. The detailed processes are introduced as follows:

First, the agent collects information from the environment, including the traffic state x_t and the model state z_t. The actor network selects an action a_t by substituting the current state S_t into the behavior policy, as in (13).

$$a_t = \mu(S_t|\theta^{\mu}) + \zeta_t \tag{13}$$

where μ is the current primary policy and ζ_t is stochastic noise.

Second, the model obtains the prediction according to the α given by a_t. We enter the next time step, update the state to S_{t+1}, and return the immediate reward r_t to the agent. Thereafter, (S_t, a_t, r_t, S_{t+1}) represents tuples input into the replay buffer as training data for the primary network.

Third, we use the mean square error (MSE) to construct the loss function of the critic network:

$$L = \frac{1}{\mathcal{N}}\sum_i(q_i - Q_{\theta^Q}(S_i, a_i)|_{a_i=\mu_{\theta^\mu}(S_i)})^2 \tag{14}$$

where q_i can be regarded as a 'label' and depends on the target network; this term is defined as follows:

$$q_i = r_i + \gamma Q'_{\theta^{Q'}}\left(s_{t+1}, \mu_{\theta^{\mu'}}(s_{i+1})\right) \tag{15}$$

The policy gradient of the critic network is shown in (14), where ε_i is the temporal-difference error.

$$\begin{cases} \nabla_{\theta^Q} L = \frac{2}{\mathcal{N}}\sum_i \varepsilon_i \cdot \nabla_{\theta^Q} Q_{\theta^Q}(S_i, a_i) \\ \varepsilon_i = q_i - Q_(\theta^Q)(S_i, a_i)|_(a_i = \mu_(\theta^\mu)(S_i)) \end{cases} \tag{16}$$

Based on the sampling transition tuples from the relay buffer and θ^Q, the actor network updates the behavior policy using (17), in which the Monte Carlo method is employed, i.e., we input minibatch-size data.

$$\nabla_{\theta^\mu} J \approx \frac{1}{\mathcal{N}}\sum_i \nabla_{\theta^\mu}\mu_{\theta^\mu}(S_i) \cdot \nabla_{a_i} Q_{\theta^Q}(S_i, a_i)|_{a_i=\mu_\theta(S_i)} \tag{17}$$

Finally, we utilize the soft updating method to partially update the parameters of the target networks by online networks, which can be formulated as follows:

$$\begin{cases} \theta^{Q'} \leftarrow \varphi\theta^Q + (1-\varphi)\theta^{Q'} \\ \theta^{\mu'} \leftarrow \varphi\theta^\mu + (1-\varphi)\theta^{\mu'} \end{cases} \tag{18}$$

where φ is an updating coefficient. When the reward converges to a stable value, we consider the optimal solution of the problem to be obtained.

3.3.3 Network structure

The structural design of the critic and actor networks is very important since they are not only function approximators but also part of feature learning. As shown in Fig. 5, the actor and critic each have a primary network and target network, and the latter can be considered a copy of the former. Therefore, they share the same network structure.

The function of actor networks is to decide the corresponding action a_t according to the state S_t, guided by the behavioral strategy β. As shown in Fig. 6, the actor networks consider the state-and-trend unit X_t and the prediction model state z_t as input and output continuous action values α of the TBSM. Considering the complex and strong time-varying characteristics of short-term traffic evolution, LSTM is applied to combine the temporal features of the state-and-trend unit as a feature extraction layer. In LSTM, each memory cell has an input gate I_t, a forget gate F_t and an output gate O_t using a $sigmoid$ activation function. They are utilized to decide whether to accept the computation results of the preceding memory cell and add them to the cell state computation (19), whether to selectively forget the preceding information from the network (20), and whether to output its own computation results (21).

$$I_t = \text{sigmoid}(w_I \cdot [\mathcal{X}_t, h_{t-1}] + b_I) \tag{19}$$

$$F_t = \text{sigmoid}(w_F \cdot [\mathcal{X}_t, h_{t-1}] + b_F) \tag{20}$$

$$O_t = \text{sigmoid}(w_O \cdot [\mathcal{X}_t, h_{t-1}] + b_O) \tag{21}$$

where w_x is the corresponding weight matrix and b_x is the bias term. $\mathcal{X}_t$ and h_t are the input and output values of the memory cell at moment t, respectively. Assume that C_t is the cell state, which can be represented as follows:

$$\begin{cases} \tilde{C}_t = \tanh(w_C \cdot [\mathcal{X}_t, h_{t-1}] + b_C) \\ C_t = C_{t-1} \otimes F_t + \tilde{C}_t \otimes I_t \\ h_t = O_t \otimes tanh(C_t) \end{cases} \tag{22}$$

The application of LSTM can effectively cope with the strong time-varying feature of short-term traffic evolution and greatly improve the accuracy of the prediction task. In addition, a rectified linear unit (ReLU) is utilized as the activation function of the fully connected layer in the network to alleviate overfitting.

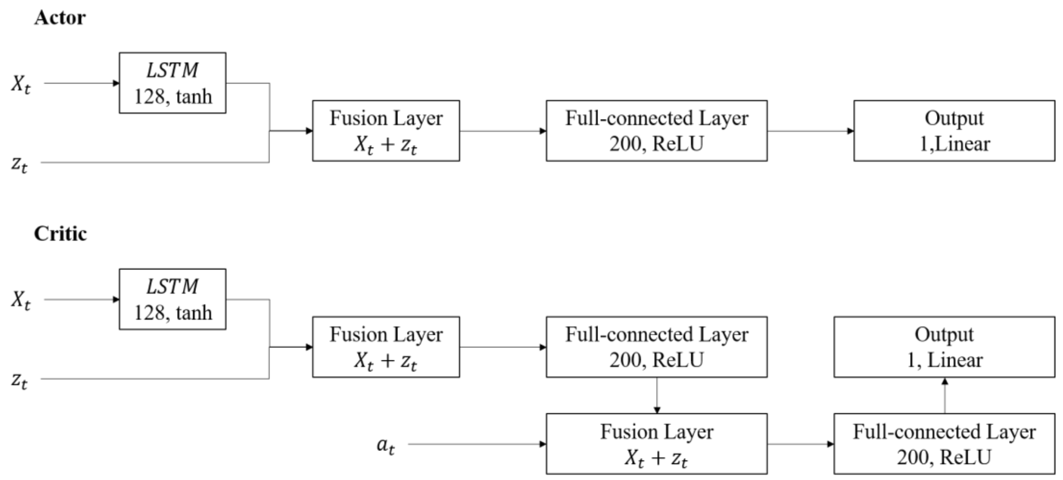


Figure 6. Structure of actor and critic networks.

The function of critic networks is to calculate the value of Q to evaluate the decisions of actor networks according to the system state S_t and action a_t. Considering that the critic network requires an additional input action value a_t, we employed a two-layer fusion layer, and the final network output layer outputs Q values through a neuron using a linear activation function, as shown in Fig. 6.

At this point, we completed the construction of the DDPG-enabled TBSM. In each time step, the trained TBSM can choose an appropriate α based on the observed state X_t and historical data $[X_{t-\delta+1}, X_{t-\delta+2}, \dots, X_{t-1}]$. This parameter α always tends to improve the prediction accuracy relative to the precalibrated model P_0. Then, the SD_i between the observed state X_t and each historical state is calculated based on α, and the nearest K of them are selected to obtain the predicted value V_{t+f} according to (10). When SE-induced traffic bursts are generated, the model can identify them and adjust itself in time to ensure the accuracy of short-term prediction. The pseudocode of the proposed method is illustrated in Algorithm 1.

```
Algorithm 1 DDPG-enabled TBSM
Input: Initial prediction model $P_0$, replay buffer $\mathscr{R} = \oslash$;
Output: Action $a_t$ fot each time step;
 1: Initialize critic network and actor network with weights $\theta^Q$ and $\theta^{\theta^\mu}$ ; the target network with
    weights $\theta^Q \leftarrow \theta^Q$ and $\theta^\mu \leftarrow \theta^\mu$ ;
 2: for episode = $1, 2, ..., M$ do
 3:   Initialize the OU stochastic process;
 4:   for t = $1, 2, ..., T$ do
 5:     Agent observes the state of environment:
            $X_t = \{V_t, \vec{\tau}_t\}, S_t = \{X_t, z_t\}$
 6:     Select action $a_t = \mu(S_t|\theta^\mu + \zeta_t)$ according to the current policy and exploration noise;
 7:     Execute action $a_t = \alpha_t$, update the prediction model $P_t$;
 8:     Obtain $z_{t+1}$ and calculate the immediate reward $r_t$ according to (12);
 9:     Obtain new state: $S_t \rightarrow S_{t+1}$ ;
10:     Store $(S_t, a_t, r_t, S_{t+1})$ in $\mathscr{R}$
11:     if sample size of replay buffer > preset value then
12:       Sample a random minibatch of $\mathscr{N}$ from $\mathscr{R}$ as a minibatch;
13:       Train the valuation networks of the actor and critic;$(S_i, a_i, r_i, S_{i+1})$ is used for a sigle tran-
          sition in $\mathscr{N}$;
14:       Define the loss function of the critic network and calculate gradient:
              $L = \frac{1}{\mathscr{N}} \sum_i (q_i - Q_{\theta^Q}(S_i, a_i)|_{a_i=\mu_{\theta^\mu}(S_i)})^2$
15:       Calculate the policy gradient of the critic network:
              $\nabla_{\theta^Q} L = \frac{2}{\mathscr{N}} \sum_i \varepsilon_i \cdot \nabla_{\theta^Q} Q_{\theta^Q}(S_i, a_i)$
16:       Calculate the behavior policy:
              $\nabla_{\theta^\mu} J \approx \frac{1}{\mathscr{N}} \sum_t \nabla_{\theta^\mu} \mu_{\theta^\mu}(S_i) \cdot \nabla_{a_i} Q_{\theta^Q}(S_i, a_i)|_{a_i=\mu_\theta(S_i)}$
17:       Update the weights using the sampled policy gradient:
              $\theta^Q \leftarrow \theta^Q - \eta \nabla_{\theta^Q} L$
              $\theta^\mu \leftarrow \theta^\mu - \eta \nabla_{\theta^\mu} J$
18:       Update the target networks according to:
              $\theta^Q \leftarrow \varphi \theta^Q + (1-\varphi)\theta^Q$
              $\theta^\mu \leftarrow \varphi \theta^\mu + (1-\varphi)\theta^\mu$
19:     end if
20:   end for
21: end for
```

4. Experiment

In this section, the proposed model is evaluated based on a real-world dataset, and the results are analyzed.

4.1 Data preparation

4.1.1 Data source

As one of the largest stadiums in Beijing, Beijing Workers' Stadium holds hundreds of large-scale cultural activities every year and therefore is very popular and influential in Beijing. Therefore, this experiment employs a proprietary traffic dataset that contains global positioning system (GPS) trajectory data of vehicles, including speed and spatiotemporal information around the Beijing Workers' stadium. This dataset is also used in [37] and [38]. The data were collected from June 30, 2015, to July 31, 2015, and the sampling frequency was 2 min (720 time steps in 1 day).

The research area is the road network around the Beijing Workers' Stadium, which encompasses 257 links and an area of 4.52 km^2. In summer, SEs are frequently held in the stadium and gymnasium. Two football games and concerts were held at Beijing Workers' Stadium and Gymnasium in July 2015, which increased the burden of this traffic network more frequently than the usual case. The dataset includes the observation of several SEs in the district of interest. We manually annotate contextual information regarding events, including the event venue, event category, event start time and end time, using information obtained via Web searches. Based on the retrieval results, the data are divided into historical, validation and test datasets at a ratio of 2:1:1, ensuring that each dataset contains SEs. Table 1 shows the dates and types of the selected samples.

Table 1. The SEs contained in the dataset.

Date	Strat	Date	Event	Dataset
Jul. 11th	19:30	21:00	Concert	Historical dataset
Jul. 12th	19:35	21:25	Football game	
Jul. 17th	19:30	21:20	Concert	Validation dataset
Jul. 20th	19:35	21:25	Football game	
Jul. 25th	19:30	21:35	Concert	Test dataset

4.1.2 Data processing

Unfavorable factors, such as outliers and missing values in original data, inevitably impact the performance of short-term traffic prediction models. In this paper, we define the time mean speed $v_l(t)$ at time t as the traffic state of link l, which can be obtained by calculating the mean speed of all cars on link l at this time, as shown in (23).

$$v_l(t) = \frac{1}{n}\sum_1^n v_i \tag{23}$$

We followed five steps to process the raw data.

1. **Data normalization**: To eliminate the influence of the urban road hierarchy, we normalize $v_l(t)$ according to the speed limit of each level as follows:

$$v_l(t) = \min\left(\frac{v_l(t)}{v_l^{\max}}, 1\right) \tag{24}$$

where n is the number of vehicles on link l at time t, v_i is the speed of each vehicle and v_l^{max} is the speed limit of road link l.

2. **Data completion**: Due to limitations from objective factors such as the data acquisition and transmission conditions, the occurrence of missing data is generally difficult to avoid. In this paper, we employ linear interpolation to complete the missing data. Specifically, the mean value of the upstream and downstream data is utilized as the filling value when missing data occur in a certain link, and the mean value of data before and after the time is used as the filling value when missing data occur at a certain time.
3. **Data Filtering**: Considering the merits of robustness to outliers, high flexibility and independence of any assumption, data denoising is carried out based on the locally weighted scatterplot smoothing (LOESS) filter to isolate the evolution trend of the traffic state. The specific processing flow refers to the past work of [48,51].
4. **Intraday trend removal**: Removing intraday trends is considered to be an effective method for improving the prediction accuracy. Concretely, all data are detrended by subtracting the average flow at the same time of the last few weeks, as suggested in [11] and [16].
5. **Data dimensionality reduction**: Due to the similarity of the traffic flow in the upstream and downstream roads, there is a strong correlation between the data characteristics of the traffic state in the complete road network. Therefore, the direct input of data leads to dimension redundancy. In this study, the Pearson correlation coefficient is used to describe the correlation of traffic state changes on each road section, and one of each pair of highly correlated links is eliminated. Assuming that the variation sequence of the traffic state on link i is l_i and that the Pearson correlation between link i and link j is calculated as shown in (25), the calculation results yield the matrix shown in Fig. 7. We employ links with high correlation $(\rho_{ij} \geqslant 0.8)$ to retain and obtain representative sections [52,53]. As a result, 94 links are selected as the input of the road network, and the entire 257 links are output.

$$\rho_{ij} = \frac{cov(l_i, l_j)}{\sqrt{D(l_i)}\sqrt{D(l_j)}} \tag{25}$$

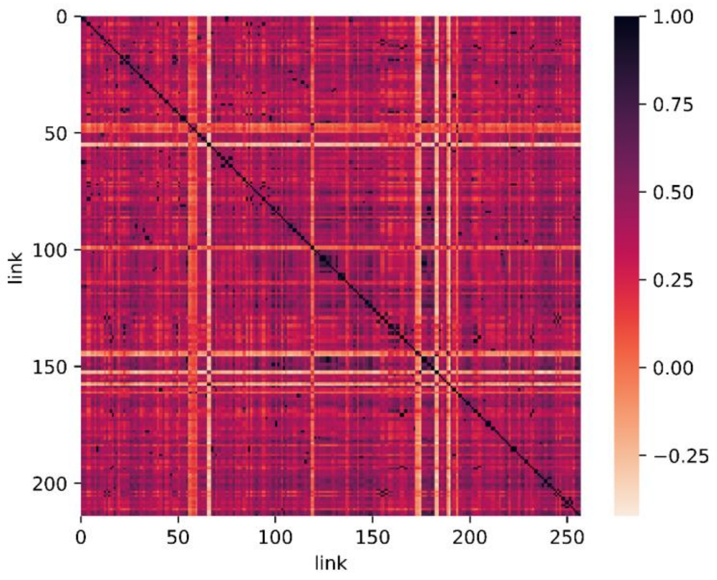


Figure 7. Correlation matrix of the links.

4.2 Experimental settings

4.2.1 Metrics

We use two metrics to evaluate the prediction performance of the proposed model.

1. Mean absolute error (MAE):

$$\mathrm{MAE} = \frac{1}{Nn}\sum_{i=1}^{N}\sum_{l=1}^{n}\left|\hat{y}_l^i - y_l^i\right| \tag{26}$$

2. Mean absolute percentage error (MAPE):

$$\mathrm{MAPE} = \frac{1}{Nn}\sum_{i=1}^{N}\sum_{l=1}^{n}\frac{\left|\hat{y}_l^i - y_l^i\right|}{y_l^i} \times 100\% \tag{27}$$

where $\hat{y}_l^i$ and y_l^i are the predicted value and actual value, respectively, at the i-th time interval in the l-th link. N denotes the number of samples in the validation or test dataset, and n denotes the number of links in the road network. In this experiment, the values of these terms are 5760 and 257, respectively.

4.2.2 Parameters

In this study, the initial parameters are calibrated by a series of basic experiments, as detailed in Appendix A. In the multistep prediction, the initial values of K and α in P_0 are listed in Table 2 and the time lag δ is set to 6.

Table 2. Parameter calibration results in each multistep prediction task.

f	K	α
1	97	0.9
2	57	0.9
3	57	0.8
4	54	0.8
5	54	0.7

The results of the precalibration reveal that K and α decrease with an increase in the prediction step f. These findings indicate that when the prediction steps increase, on the one hand, traffic states are more variable because of longer time-varying processes, and thus, it is more difficult to capture similar state units. On the other hand, the impact of the benchmark state weakens gradually, and thus, the distance metric is supposed to decrease the proportion of the benchmark state and increase the proportion of the trend vector by means of adjusting the value of α

4.2.3 Benchmark methods

Four mainstream machine learning models and two advanced DL models are employed as the benchmark methods. The implementation details are introduced briefly as follows:

- **KNN**: The original KNN method uses the ED as the similarity measure and takes the mean value of neighbor tags as the prediction result. K=14.
- **SVR**: SVR is a widely used machine learning model, and the parameters are determined with the kernel of the Gaussian radial basis function, where V=10 and x=0.01.
- **RF**: The random forest (RF) is a flexible and stable machine learning model based on the integration of multiple decision trees and the bagging algorithm. After numerical experiment calibration, we use the Scikit-Learn toolkit to build an RF comparison model with 100 decision trees.
- **GBDT:** The gradient boosted decision tree (GBDT) is an ensemble model of decision trees based on a boosting algorithm. The optimized parameter is adopted in experiments by the grid search method. The number of decision trees is 100; the depth of the trees is 5; and the learning rate is 0.01.
- **SAE**: The SAE is a depth structure for short-term traffic state prediction that adopts greedy layerwise pretraining. The SAE structure constructed in this paper is shown in Appendix B.
- **SRCN**: The SRCN has achieved a better prediction performance through the deep combination modeling of a CNN and LSTM. To ensure the intended effect, this paper uses the same processing method as the original. The details are shown in Appendix B.

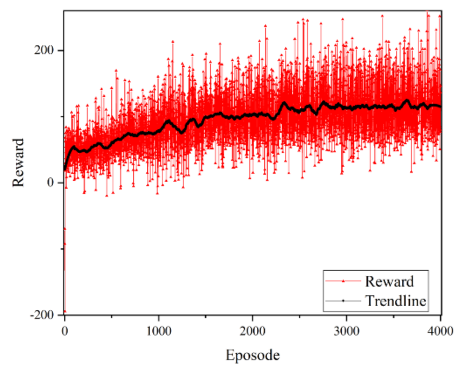
4.2.4 Computational process

The DNNs in the actor-critic network are based on the Keras neural network library with the TensorFlow framework as the back end, and the optimizer is adaptive moment estimation (Adam). Experiments to evaluate the proposed model include one to five step-ahead predictions. All of these experiments are performed on a computing platform with an Intel Core i7-9700K CPU (3.60 GHz), NVIDIA GeForce RTX 2080Ti graphics processing unit (GPU), with 32.0 GB memory.

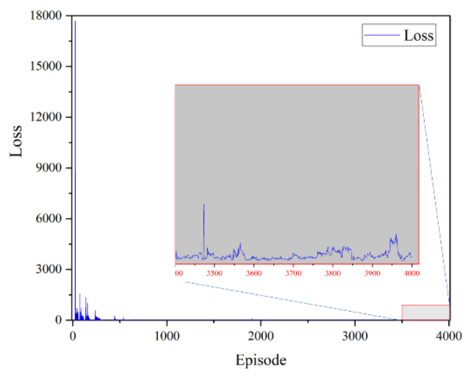
4.3 Results

4.3.1 Training phase

The reward and round average loss function values are usually used to reflect the training convergence level and learning effect. Fig. 8(a) shows that the reward increases with the episode. In the early stage of training, the cumulative rewards of the round are negative, and with the progress of the training, positive promotion occurs. Additionally, the average loss of the critic network decreases with the number of rounds of training and falls below 0.4, as shown in Fig. 8(b). It is apparent that both objective values of the two networks converge as the training iteration increases.



(a) Reward in each episode during the training phase.



(b) Average loss of the critic network.

Figure 8. Evolution of reward and average loss during the training phase.

4.3.2 Overall evaluation

Table 3 shows the overall performance of the proposed model and other baseline methods for the prediction step ranging from 1 to 5, and the improvement rate is shown in Table 4. The corner mark * means that the value is optimal. It is clear that the TBSM shows optimal performance in short-term prediction ($f = 1,2,3$) under the MAE and MAPE criteria. In particular, when $f = 1$, it achieves a test error rate (MAPE) of 11.1%, which is almost 60% better than these conventional models.

Table 3. Prediction error of the TBSM and the other baseline methods.

Models	$f = 1$		$f = 2$		$f = 3$		$f = 4$		$f = 5$	
	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE
TBSM	**2.48***	**11.10***	**4.13***	**18.82***	**4.84***	**22.44***	5.16	**24.35***	5.34	**25.52***
KNN	4.74	25.84	4.90	26.74	5.06	27.80	**5.14***	28.41	**5.19***	28.82
SVR	4.17	21.31	5.06	26.07	5.30	27.60	5.42	28.37	5.48	28.99
RF	5.31	29.13	5.37	29.66	5.41	30.10	5.42	30.26	5.43	30.52
GBDT	3.99	23.79	4.79	27.97	5.08	29.80	5.21	30.80	5.29	31.42
SAE	5.01	26.29	5.04	27.94	5.07	28.16	5.15	28.22	5.21	29.11
SRCN	4.92	26.51	5.00	27.56	5.12	29.04	5.22	29.32	5.20	29.85

In addition, it can be observed that the advantage of the TBSM gradually narrows with an increase in the number of prediction steps. When $f = 4$ and 5, although the TBSM still shows relatively good performance in terms of MAPE, it does not differ considerably from the other models in terms of MAE. This can be explained as follows: on the one hand, with the increase in f, it was indeed difficult to predict because of the more complex and uncertain variation; on the other hand, the state-and-trend units constructed in the TBSM are more focused on capturing short-term trends and should be updated for traffic forecasts with longer steps.

Table 4. The improvement of TBSM over benchmark methods.

Models	$f = 1$		$f = 2$		$f = 3$		$f = 4$		$f = 5$	
	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE	MAE	MAPE
TBSM	-	-	-	-	-	-	-	-	-	-
KNN	47.68%	57.04%	15.71%	29.62%	4.35%	19.28%	-0.39%	14.29%	-2.89%	11.45%
SVR	40.53%	47.91%	18.38%	27.81%	8.68%	18.70%	4.80%	14.17%	2.55%	11.97%
RF	53.30%	61.89%	23.09%	36.55%	10.54%	25.45%	4.80%	19.53%	1.66%	16.38%
GBDT	37.84%	53.34%	13.78%	32.71%	4.72%	24.70%	0.96%	20.94%	-0.95%	18.78%
SAE	50.50%	57.78%	18.06%	32.64%	4.54%	20.31%	-0.19%	13.71%	-2.50%	12.33%
SRCN	49.59%	58.13%	17.40%	31.71%	5.47%	22.73%	1.15%	16.95%	-2.69%	14.51%

Another noteworthy consideration is that DL models, i.e., SAE and SRCN, seemed mediocre for this issue, although it has been indicated that high accuracy can be obtained in short-term traffic state prediction tasks under various scenarios. A possible reason is that the DL model is highly dependent on a large number of training samples and high-quality data, both of which are scarce in the SE scenario. Simultaneously, the construction mechanism of the DL model attempts to minimize the error between its output value and the real labeled value, and training with a mixture of data from the evolution of conventional traffic states and the evolution of traffic states under SE characteristics may produce a model that occupies the middle ground between the two scenarios, and this construction method causes the prediction accuracy to suffer in both scenarios.

4.3.3 Performance in traffic bursts

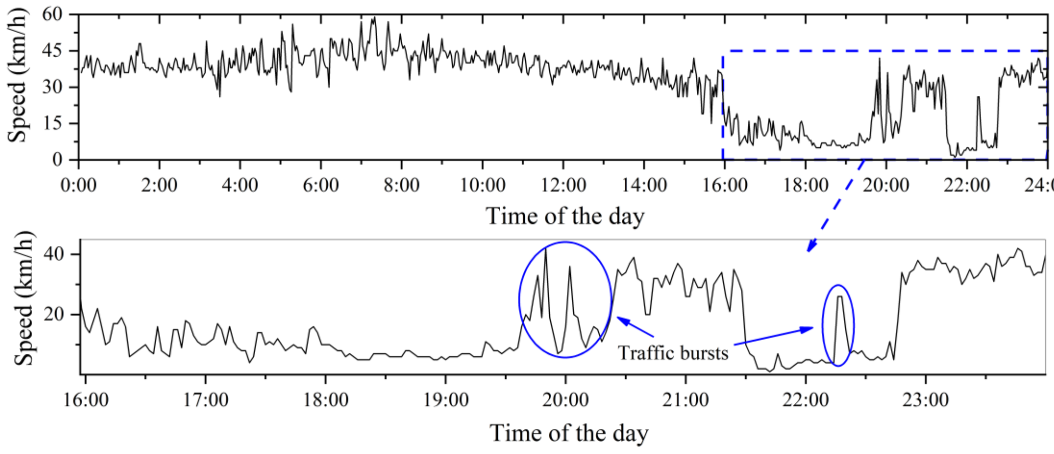


Figure 9. Average speed of link-12 on July 25th (with a concert).

To further illustrate the performance of the TBSM under SEs, the performance of each model is analyzed by considering the evolution and prediction of the traffic state of a certain section of the West Road of Worker's Stadium (link-12) as an example. There was a concert that took place in this area between 19:30 and 21:35 on July 25, 2015. Its real traffic state changes are shown in Fig. 9. In the afternoon of that day, the average speed of the section showed a relatively obvious decline, and the low speed lasted for a long time before the concert started (approximately 19:30).

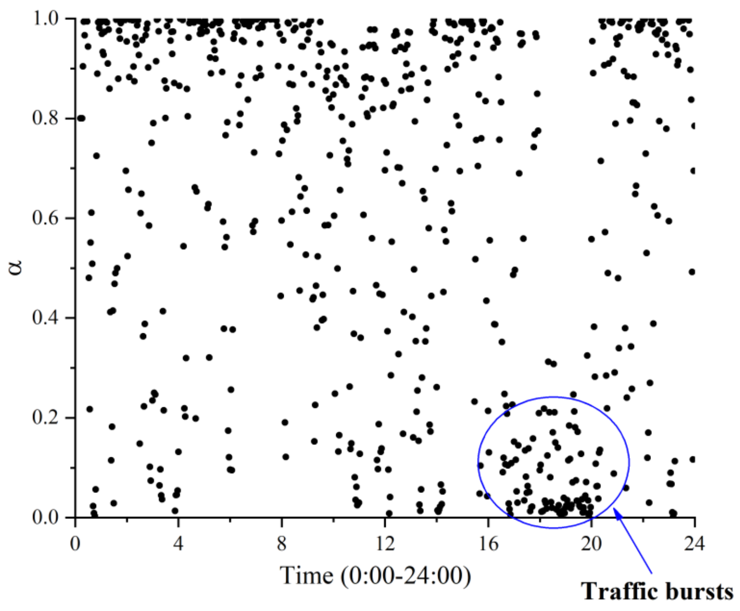


Figure 10. Action of the TBSM in single-step prediction under a SE.

We observe bursts occurring from approximately 19:30 to 21:30. In response, in single-step prediction, the trained agent chooses α values approaching 0, which makes the prediction model more sensitive to the trend, as shown in Fig. 10. Fig. 11 shows the visualization of traffic forecasting on that day. Our TBSM captures almost all bursts, and the results are excellent, which means that the agent can meet our expectations; i.e., it can select the actions with the highest predicted accuracy for different observation states.

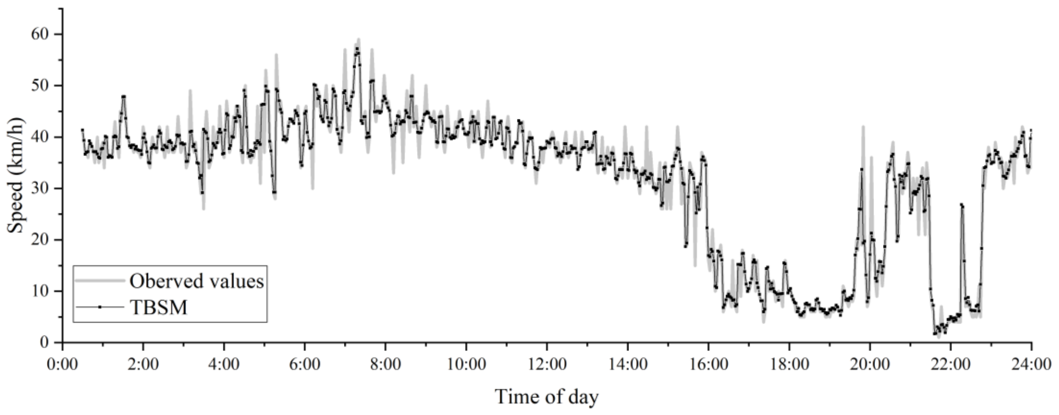


Figure 11. Performance of the TBSM in single-step prediction under a SE.

Residuals between 16:00 and 24:00 are selected for observation of specific prediction deviations, as shown in Fig. 12. In the face of bursts caused by a sharp increase in traffic demand, the prediction model often exhibits peak error at this time. In the short-term prediction ($f = 1,2$), the predicted residual of the TBSM is stable at a low level, which proves its superiority. With increasing f, the residual values predicted by each model fluctuate greatly, especially in the first hour after the concert (21:35-22:25). In the fluctuation of the prediction effect during this period, our model can be adjusted the fastest, and the residuals decline the most rapidly, showing more stable and reliable abilities than the other models.

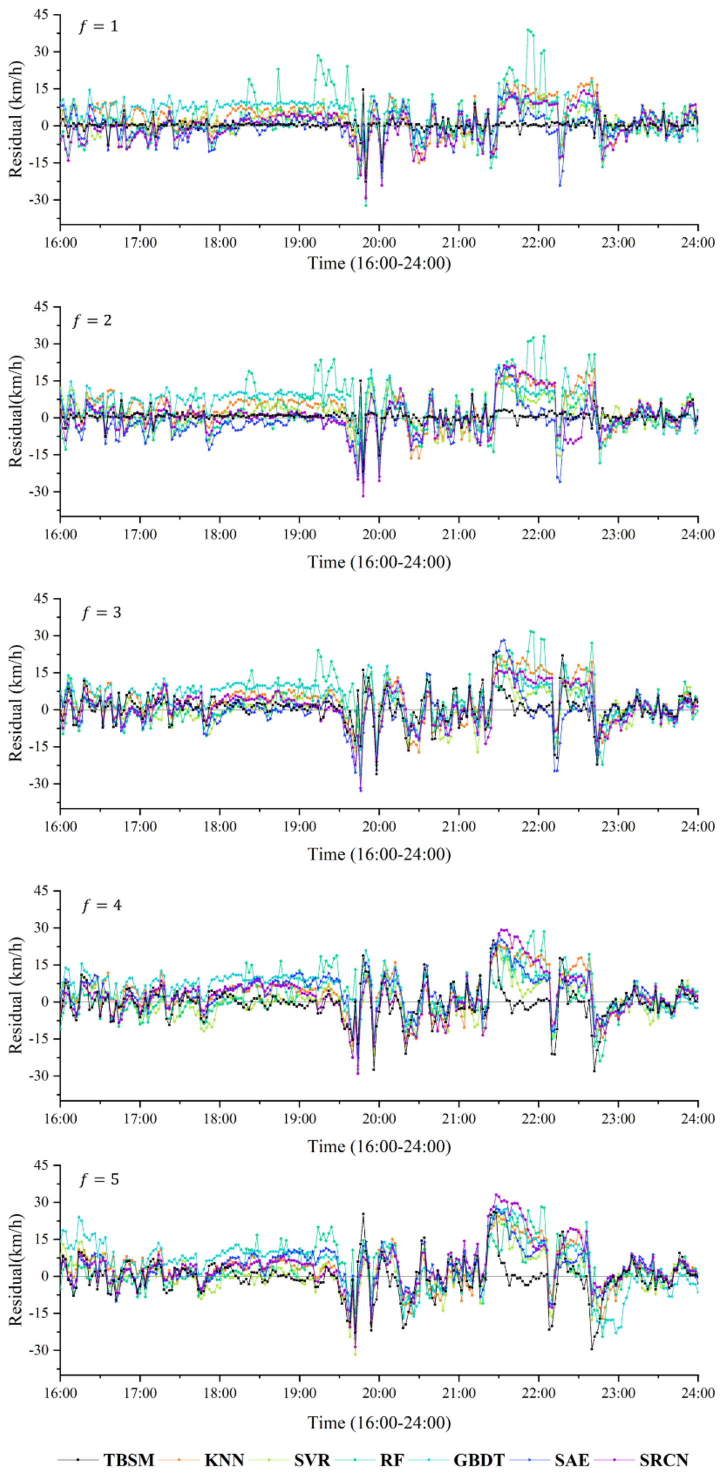


Figure 12. Prediction residuals of four prominent models after the concert.

As a relative indicator, the MAPE can magnify the difference between the predicted and true values when the former is small. Therefore, the MAPE has more reference value in the low-speed state after the concert. In Fig. 13, we select four models with better performance in the overall evaluation to observe the changes in their MAPE values between 21:00 and 23:00. Compared to other models, especially the DL models, our TBSM (the black line) shows the lowest peak error and the fastest adjustment speed in all prediction step sizes, suggesting that it is more adaptable and accurate in short-term traffic prediction under SEs.

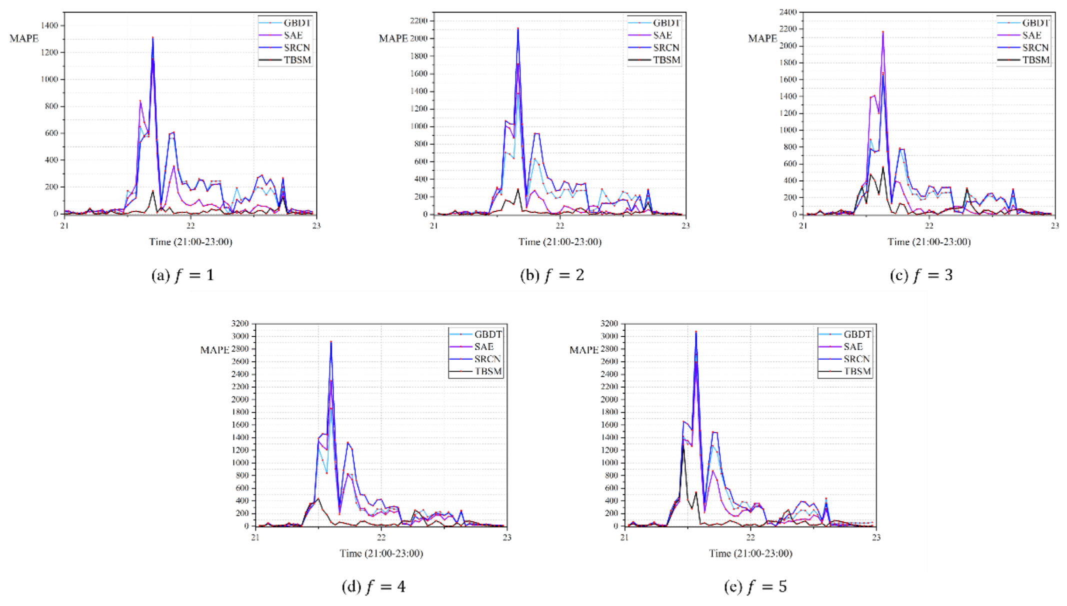


Figure 13. MAPE values of four prominent models (GBDT, SAE, SRCN and TBSM) after the concert.

4.3.4 Statistical tests

To assess the accuracy of multistep forecasting, we analyze the residuals, namely, the difference between observed values and the predicted values. References [51] and [54] were mainly referred to for relevant experimental settings. The results of the Kolmogorov–Smirnov test suggest that the prediction residuals of all models do not follow a normal distribution, as shown in Table 5 [55]. Therefore, a nonparametric statistical hypothesis test, the Wilcoxon signed-rank test, is employed to further check if the proposed model is statistically better than all baselines [56].

In this experimental setting, we use MAE instead of residuals to avoid negative and positive differences. The null hypothesis for the Wilcoxon signed-rank test is that there is no difference in the forecast error between the proposed methods and baselines, while the alternative hypothesis is that relative to baseline methods, the reduction in the forecast error of the proposed model is greater than zero. The results in Table 6 indicate that when $f = 1{,}2$, the reductions in forecast errors are found to be very statistically significant ($P < 0.001$), and thus, the null hypothesis is rejected, i.e., there are statistically significant differences between our method and all of the comparison methods. When $f = 3$, except for SAE, the TBSM is significantly superior to any other model. When $f = 4, 5$, although TBSM leads by a small margin with regard to the MAE and MAPE, it is not statistically optimal.

Based on the above analysis, the developed model shows a significant improvement in performance compared to other models in short-term prediction (2 min, 4 min and 6 min).

Table 5. Results of the Kolmogorov–Smirnov test for the distribution: values of the corresponding test statistic and p value.

Kolmogorov–Smirnov Test	$f = 1$		$f = 2$		$f = 3$		$f = 4$		$f = 5$	
	Z	P	Z	P	Z	P	Z	P	Z	P
TBSM	0.208	0	0.179	0	0.079	0	0.076	0	0.074	0
KNN	0.079	0	0.087	0	0.042	0.005	0.076	0	0.101	0
SVR	0.138	0	0.103	0	0.107	0	0.106	0	0.12	0
RF	0.045	0.001	0.071	0	0.088	0	0.066	0	0.072	0
GBDT	0.056	0	0.044	0.002	0.046	0.001	0.055	0	0.065	0
SAE	0.111	0	0.09	0	0.092	0	0.084	0	0.089	0
SRCN	0.055	0	0.046	0.001	0.058	0	0.062	0	0.074	0

Table 6. Results of the Wilcoxon signed-rank test. The table shows the value of the corresponding test statistic and p value.

Wilcoxon Signed-Rank Test	$f = 1$		$f = 2$		$f = 3$		$f = 4$		$f = 5$	
	Z	P	Z	P	Z	P	Z	P	Z	P
TBSM	-20.246	0	-19.393	0	-5.509	0	-2.347	0.019	-0.546	0.585
KNN	-20.274	0	-20.343	0	-10.685	0	-8.809	0	-8.879	0
SVR	-18.589	0	-18.142	0	-1.396	0.16	-0.656	0.512	-0.204	0.838
RF	-18.415	0	-18.897	0	-4.243	0	-1.704	0.088	-1.078	0.281
GBDT	-19.576	0	-19.694	0	-4.3337	0	-1.168	0.243	-0.329	0.742
SAE	-20.084	0	-20.157	0	-5.233	0	-2.073	0.038	-0.766	0.443
SRCN	-20.246	0	-19.393	0	-5.509	0	-2.347	0.019	-0.546	0.585

5. Conclusion

Due to the uncontrollable elements of SEs, it is difficult to obtain abundant data and the desired prediction accuracy under such conditions. In this paper, we present the TBSM to address this challenge. Different from the previous work, we construct state-and-trend units to capture traffic bursts and accordingly develop a novel expression for sample similarity. Furthermore, we propose an incremental prediction form and transform the short-term prediction problem into a parameter optimization problem based on historical data and observation states by Gaussian weighting. A DDPG framework with an LSTM is employed to dynamically adjust the model to solve this problem and realize generality and real-time behavior. The self-learning nature of RL can help the TBSM eliminate reliance on large amounts of labeled data while making accurate predictions.

Note that the prediction of traffic flow under SEs is evidently more challenging than doing so under usual cases and, hence, much desired by operational agencies. The experimental results on a real-world dataset demonstrate that our proposed model can capture traffic bursts caused by SEs and significantly outperforms all baselines in short-term prediction (2 min, 4 min and 6 min). That is, the TBSM is found to be suitable and useful in real-world operations, for example, to guide traffic signal control after a concert and adjust evacuation plans for sports events.

One limitation of this study is that the model is not as efficient for high-dimensional input. In future studies, distributed execution and multiagent strategies will be taken into account.

Acknowledgments

The authors are grateful to anonymous referees whose comments greatly improved the quality of this article. This study was supported by the National Natural Science Foundation of China (No. U1811463, 51908018 and 51878020), China Postdoctoral Science Foundation funded project (Grant No. 2021M690296).

Appendix A. Hyperparameter search

To accelerate the training and convergence of the model, the parameters of the initial model P_0 need to be calibrated at each prediction step. The hyperparameters in the TBSM include the prediction time lag δ, the number of nearest neighbors K and the equilibrium factor α.

Appendix A.1 Time lagδ

The time lag δ is regarded as an important parameter to determine the input of the proposed model. It is generally recognized that significant changes in traffic states can occur within a few minutes. To determine which value of δ achieves the optimal performance, we observed the influence of different l values in a wider range with a maximum of 20 min ($\delta = 1,2,...,10$) and discovered that $\delta = 6$ has the least influence on MAE and MAPE. For example, Fig. A-14 shows the performance results when $f = 2$ and other parameters are fixed.

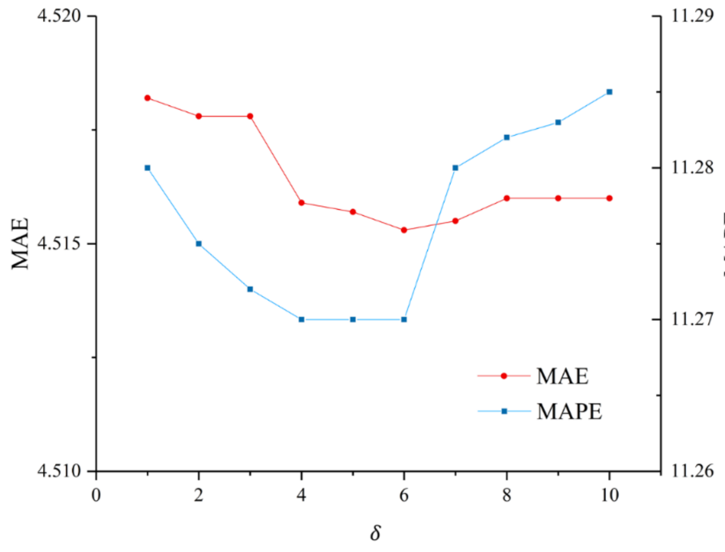
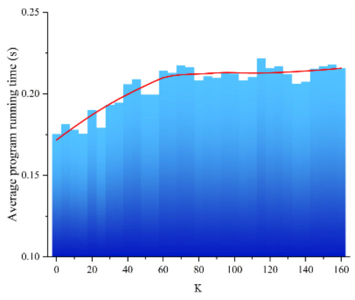


Figure A-14. Influence of δ on MAE and MAPE of the model when $f = 2$.

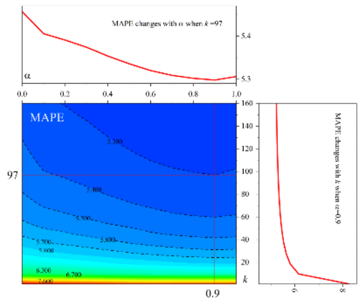
Appendix A.2 K andα

Considering the small number of hyperparameters contained in the TBSM, we use the grid search method to determine the values of K and α in the initial model P_0 [57]. The core idea is to construct and train a model for each pair of parameter combinations in the Cartesian product of the set of different parameter values and to evaluate the predictive effect of the resulting model on the validation set. This kind of exhaustive search optimization is straightforward but very useful and can be implemented through the Scikit-Learn package.

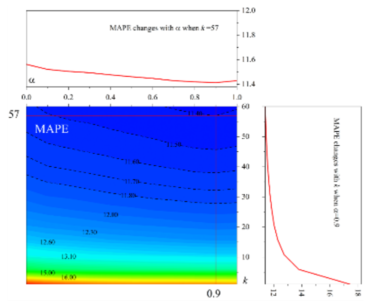
In the experiments, the influence of both K and α on the root mean square error (RMSE), MAE and MAPE are taken into account, and we finally chose MAPE, which is the most sensitive to parameters, for reference. Fig. A-15 (b) to Fig. A-15 (f) show the results, and Fig. A-15 (a) shows the average program running time for each K value. The optimal parameter combination can be determined through the dotted line in the thermal diagram. For example, for a single-step prediction ($f = 1$), the optimal value of K is 97, and α is 0.9 to obtain the minimum MAPE. In this way, we obtain the parameter values of P_0 for each prediction step, as listed in Table 2.



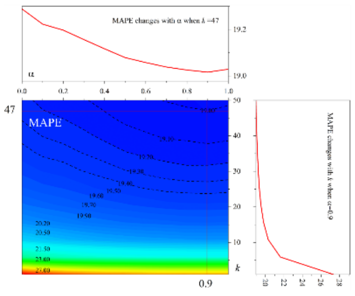
(a) Average program running time (s)



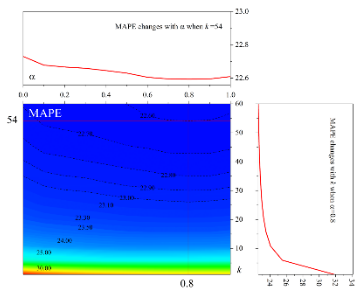
(b) MAPE, $f = 1$



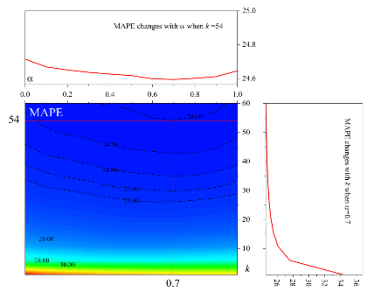
(c) MAPE, $f = 2$



(d) MAPE, $f = 3$



(e) MAPE, $f = 4$



(f) MAPE, $f = 5$

Figure A-15. Influence of K and α on prediction performance in grid search.

Appendix B. Details of neural network hyperparameters in SAE and SRCN

We developed an SAE model according to the literature [36]. We added the dropout layer to prevent overfitting before the final output layer. The details of the SAE used in this paper are shown in Table B.7.

Table B.7. Details of the SAE architecture.

Layer	Type	Number of units
0	Input	6×257
1	Hidden layer 1	400
	ReLU	
2	Hidden layer 2	400
	ReLU	
3	Hidden layer 3	400
	ReLU	
4	Dropout	0.2
5	Output	257

We developed an SRCN model according to the literature [37]. Two dropout layers were employed to prevent overfitting. Batch normalization was applied to accelerate training. The details of SRCN used in this paper are shown in Table B.8.

Table B.8. Details of the SRCN architecture.

Layer	Type	Channels	Size
0	Input	1	162×224
1	Convolution 1	16	(3,3)
2	Max-pooling 1	16	(2,2)
	ReLU		
	Batch normalization		
3	Convolution 2	32	(3,3)
4	Max-pooling 2	32	(2,2)
	ReLU		
	Batch normalization		
5	Convolution 3	64	(3,3)
	ReLU		
	Batch normalization		
6	Convolution 4	64	(3,3)
	ReLU		
	Batch normalization		
7	Convolution 5	128	(3,3)
8	Max-pooling 3	128	(2,2)
	ReLU		
	Batch normalization		
9	Faltten		
10	Full connection		257
11	LSTM 1	1	800
	Tanh		
12	Dropout		0.2
13	LSTM 2	1	800
	Tanh		
14	Dropout		0.2
15	Output	1	257

References

[1] J. Zhang, F.-Y. Wang, K. Wang, W.-H. Lin, X. Xu, C. Chen, Data-driven intelligent transportation systems: A survey, IEEE Transactions on Intelligent Transportation Systems. 12 (2011) 1624–1639.

[2] E.I. Vlahogianni, M.G. Karlaftis, J.C. Golias, Short-term traffic forecasting: Where we are and where we're going, Transportation Research Part C: Emerging Technologies. 43 (2014) 3–19.

[3] D. Yu, C. Liu, Y. Wu, S. Liao, T. Anwar, W. Li, C. Zhou, Forecasting short-term traffic speed based on multiple attributes of adjacent roads, Knowledge-Based Systems. 163 (2019) 472–484.

[4] R. Vinayakumar, K. Soman, P. Poornachandran, Applying deep learning

approaches for network traffic prediction, in: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2017: pp. 2353–2358.
[5] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, Y. Wang, Learning traffic as images: a deep convolutional neural network for large-scale transportation network speed prediction, Sensors. 17 (2017) 818.
[6] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, H. Li, T-gcn: A temporal graph convolutional network for traffic prediction, IEEE Transactions on Intelligent Transportation Systems. 21 (2019) 3848–3858.
[7] A.-L. Barabasi, The origin of bursts and heavy tails in human dynamics, Nature. 435 (2005) 207–211.
[8] H. Yu, N. Ji, Y. Ren, C. Yang, A special event-based K-nearest neighbor model for short-term traffic state prediction, Ieee Access. 7 (2019) 81717–81729.
[9] S. Latoski, W. Dunn, Managing travel for planned special events handbook, Department of Transportation, Washington DC. (2003).
[10]C. Chen, Y. Wang, L. Li, J. Hu, Z. Zhang, The retrieval of intra-day trend and its influence on traffic prediction, Transportation Research Part C: Emerging Technologies. 22 (2012) 103–118.
[11]Y. Lin, X. Dai, L. Li, F.-Y. Wang, Pattern sensitive prediction of traffic flow based on generative adversarial framework, IEEE Transactions on Intelligent Transportation Systems. 20 (2018) 2395–2400.
[12]L. Li, X. Su, Y. Zhang, Y. Lin, Z. Li, Trend modeling for traffic time series analysis: An integrated study, IEEE Transactions on Intelligent Transportation Systems. 16 (2015) 3430–3439.
[13]F. Zhou, Q. Yang, K. Zhang, G. Trajcevski, T. Zhong, A. Khokhar, Reinforced Spatiotemporal Attentive Graph Neural Networks for Traffic Forecasting, IEEE Internet of Things Journal. 7 (2020) 6414–6428.
[14]M. Ni, Q. He, J. Gao, Using social media to predict traffic flow under special event conditions, in: The 93rd Annual Meeting of Transportation Research Board, 2014.
[15]R. Yu, Y. Li, C. Shahabi, U. Demiryurek, Y. Liu, Deep learning: A generic approach for extreme condition traffic forecasting, in: Proceedings of the 2017 SIAM International Conference on Data Mining, SIAM, 2017: pp. 777–785.
[16]X. Dai, R. Fu, Y. Lin, L. Li, F.-Y. Wang, Deeptrend: A deep hierarchical neural network for traffic flow prediction, ArXiv Preprint ArXiv:1707.03213. (2017).
[17]S. Feng, X. Wang, H. Sun, Y. Zhang, L. Li, A better understanding of long-range temporal dependence of traffic flow time series, Physica A: Statistical Mechanics and Its Applications. 492 (2018) 639–650.
[18]D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, M. Riedmiller, Deterministic policy gradient algorithms, in: International Conference on Machine Learning, PMLR, 2014: pp. 387–395.
[19]D.A. Tedjopurnomo, Z. Bao, B. Zheng, F. Choudhury, A. Qin, A survey on modern deep neural network for traffic prediction: Trends, methods and challenges, IEEE Transactions on Knowledge and Data Engineering. (2020).
[20]X. Xu, J. Liu, H. Li, J.-Q. Hu, Analysis of subway station capacity with the use of

queueing theory, Transportation Research Part C: Emerging Technologies. 38 (2014) 28–43.
[21]Q. Wang, L. Li, J. Hu, B. Zou, Traffic velocity distributions for different spacings, Journal of Tsinghua University Science and Technology. 51 (2011) 309–312.
[22]P. Wei, Y. Cao, D. Sun, Total unimodularity and decomposition method for large-scale air traffic cell transmission model, Transportation Research Part B: Methodological. 53 (2013) 1–16.
[23]K. Aboudolas, M. Papageorgiou, E. Kosmatopoulos, Store-and-forward based methods for the signal control problem in large-scale congested urban road networks, Transportation Research Part C: Emerging Technologies. 17 (2009) 163–174. https://doi.org/10.1016/j.trc.2008.10.002.
[24]Z. Shan, D. Zhao, Y. Xia, Urban road traffic speed estimation for missing probe vehicle data based on multiple linear regression model, in: 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), 2013: pp. 118–123. https://doi.org/10.1109/ITSC.2013.6728220.
[25]L. Li, L. Qin, X. Qu, J. Zhang, Y. Wang, B. Ran, Day-ahead traffic flow forecasting based on a deep belief network optimized by the multi-objective particle swarm algorithm, Knowledge-Based Systems. 172 (2019) 1–14. https://doi.org/10.1016/j.knosys.2019.01.015.
[26]D. Xia, H. Li, B. Wang, Y. Li, Z. Zhang, A Map Reduce-Based Nearest Neighbor Approach for Big-Data-Driven Traffic Flow Prediction, IEEE Access. 4 (2016) 2920–2934. https://doi.org/10.1109/ACCESS.2016.2570021.
[27]A.M. Nagy, V. Simon, Survey on traffic prediction in smart cities, Pervasive and Mobile Computing. 50 (2018) 148–163.
[28]L. Zhu, F.R. Yu, Y. Wang, B. Ning, T. Tang, Big data analytics in intelligent transportation systems: A survey, IEEE Transactions on Intelligent Transportation Systems. 20 (2018) 383–398.
[29]M.M. Hamed, H.R. Al-Masaeid, Z.M.B. Said, Short-term prediction of traffic volume in urban arterials, Journal of Transportation Engineering. 121 (1995) 249–254.
[30]M. Van Der Voort, M. Dougherty, S. Watson, Combining Kohonen maps with ARIMA time series models to forecast traffic flow, Transportation Research Part C: Emerging Technologies. 4 (1996) 307–318.
[31]S. Lee, D.B. Fambro, Application of subset autoregressive integrated moving average model for short-term freeway traffic volume forecasting, Transportation Research Record. 1678 (1999) 179–188.
[32]B.M. Williams, L.A. Hoel, Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results, Journal of Transportation Engineering. 129 (2003) 664–672.
[33]W. Huang, G. Song, H. Hong, K. Xie, Deep architecture for traffic flow prediction: deep belief networks with multitask learning, IEEE Transactions on Intelligent Transportation Systems. 15 (2014) 2191–2201.
[34]Z. Zhao, W. Chen, X. Wu, P.C. Chen, J. Liu, LSTM network: a deep learning approach for short-term traffic forecast, IET Intelligent Transport Systems. 11 (2017) 68–75.

[35] X. Ma, Z. Tao, Y. Wang, H. Yu, Y. Wang, Long short-term memory neural network for traffic speed prediction using remote microwave sensor data, Transportation Research Part C: Emerging Technologies. 54 (2015) 187–197.
[36] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang, Traffic flow prediction with big data: a deep learning approach, IEEE Transactions on Intelligent Transportation Systems. 16 (2014) 865–873.
[37] H. Yu, Z. Wu, S. Wang, Y. Wang, X. Ma, Spatiotemporal recurrent convolutional networks for traffic prediction in transportation networks, Sensors. 17 (2017) 1501.
[38] G. Yang, Y. Wang, H. Yu, Y. Ren, J. Xie, Short-term traffic state prediction based on the spatiotemporal features of critical road sections, Sensors. 18 (2018) 2287.
[39] L. Chen, L. Wu, R. Hong, K. Zhang, M. Wang, Revisiting graph based collaborative filtering: A linear residual graph convolutional network approach, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020: pp. 27–34.
[40] J. Zhang, F. Chen, Z. Cui, Y. Guo, Y. Zhu, Deep learning architecture for short-term passenger flow forecasting in urban rail transit, IEEE Transactions on Intelligent Transportation Systems. (2020).
[41] J. Ye, J. Zhao, K. Ye, C. Xu, How to build a graph-based deep learning architecture in traffic domain: A survey, IEEE Transactions on Intelligent Transportation Systems. (2020).
[42] C. Zheng, X. Fan, C. Wang, J. Qi, Gman: A graph multi-attention network for traffic prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020: pp. 1234–1241.
[43] K. Guo, Y. Hu, Z. Qian, H. Liu, K. Zhang, Y. Sun, J. Gao, B. Yin, Optimized graph convolution recurrent neural network for traffic prediction, IEEE Transactions on Intelligent Transportation Systems. (2020).
[44] Z. Cui, R. Ke, Z. Pu, X. Ma, Y. Wang, Learning traffic as a graph: A gated graph wavelet recurrent neural network for network-scale traffic prediction, Transportation Research Part C: Emerging Technologies. 115 (2020) 102620.
[45] X. Dai, R. Fu, E. Zhao, Z. Zhang, Y. Lin, F.-Y. Wang, L. Li, DeepTrend 2.0: A light-weighted multi-scale traffic prediction model using detrending, Transportation Research Part C: Emerging Technologies. 103 (2019) 142–157.
[46] L. Zhang, Q. Liu, W. Yang, N. Wei, D. Dong, An improved k-nearest neighbor model for short-term traffic flow prediction, Procedia-Social and Behavioral Sciences. 96 (2013) 653–662.
[47] X. Zhu, C. Ying, J. Wang, J. Li, X. Lai, G. Wang, Ensemble of ML-KNN for classification algorithm recommendation, Knowledge-Based Systems. 221 (2021) 106933.
[48] R.M. Aliguliyev, Performance evaluation of density-based clustering methods, Information Sciences. 179 (2009) 3583–3602.
[49] P. Cai, Y. Wang, G. Lu, P. Chen, C. Ding, J. Sun, A spatiotemporal correlative k-nearest neighbor model for short-term traffic multistep forecasting, Transportation Research Part C: Emerging Technologies. 62 (2016) 21–34.
[50] S. Luo, X. Lin, Z. Zheng, A novel CNN-DDPG based AI-trader: Performance and roles in business operations, Transportation Research Part E: Logistics and

Transportation Review. 131 (2019) 68–79.
[51] F.G. Habtemichael, M. Cetin, Short-term traffic flow rate forecasting based on identifying similar traffic patterns, Transportation Research Part C: Emerging Technologies. 66 (2016) 61–78.
[52] A. Guitton, A. Skordylis, N. Trigoni, Utilizing correlations to compress time-series in traffic monitoring sensor networks, in: 2007 IEEE Wireless Communications and Networking Conference, IEEE, 2007: pp. 2479–2483.
[53] S. Yang, On feature selection for traffic congestion prediction, Transportation Research Part C: Emerging Technologies. 26 (2013) 160–169.
[54] N.G. Polson, V.O. Sokolov, Deep learning for short-term traffic flow prediction, Transportation Research Part C: Emerging Technologies. 79 (2017) 1–17.
[55] F.J. Massey Jr, The Kolmogorov-Smirnov test for goodness of fit, Journal of the American Statistical Association. 46 (1951) 68–78.
[56] R. Woolson, Wilcoxon signed-rank test, Wiley Encyclopedia of Clinical Trials. (2007) 1–3.
[57] P. Lerman, Fitting segmented regression models by grid search, Journal of the Royal Statistical Society: Series C (Applied Statistics). 29 (1980) 77–84.