Computer Networks 181 (2020) 107530

Contents lists available at ScienceDirect

Computer Networks

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

Survey paper

Machine Learning-based traffic prediction models for Intelligent Transportation Systems

Azzedine Boukerche, Jiahao Wang *

Paradise Research Laboratory, School of Electrical Engineering and Computer Science, University of Ottawa, 800 King Edward Ave., Ottawa, K1N 6N5 ON, Canada

ARTICLE INFO

Keywords:
Vehicular traffic flow prediction
Time-series
GCN
Parallel training
RNN
Sequence to sequence
Machine Learning
ITS

ABSTRACT

Intelligent Transportation Systems (ITS) have attracted an increasing amount of attention in recent years. Thanks to the fast development of vehicular computing hardware, vehicular sensors and citywide infrastructures, many impressive applications have been proposed under the topic of ITS, such as Vehicular Cloud (VC), intelligent traffic controls, etc. These applications can bring us a safer, more efficient, and also more enjoyable transportation environment. However, an accurate and efficient traffic flow prediction system is needed to achieve these applications, which creates an opportunity for applications under ITS to deal with the possible road situation in advance. To achieve better traffic flow prediction performance, many prediction methods have been proposed, such as mathematical modeling methods, parametric methods, and non-parametric methods. Among the non-parametric methods, the one of the most famous methods today is the Machine Learning-based (ML) method. It needs less prior knowledge about the relationship among different traffic patterns, less restriction on prediction tasks, and can better fit non-linear features in traffic data. There are several sub-classes under the ML method, such as regression model, kernel-based model, etc. For all these models, it is of vital importance that we choose an appropriate type of ML model before building up a prediction system. To do this, we should have a clear view of different ML methods; we investigate not only the accuracy of different models, but the applicable scenario and sometimes the specific type of problem the model was designed for. Therefore, in this paper, we are trying to build up a clear and thorough review of different ML models, and analyze the advantages and disadvantages of these ML models. In order to do this, different ML models will be categorized based on the ML theory they use. In each category, we will first give a short introduction of the ML theory they use, and we will focus on the specific changes made to the model when applied to different prediction problems. Meanwhile, we will also compare among different categories, which will help us to have a macro overview of what types of ML methods are good at what types of prediction tasks according to their unique model features. Furthermore, we review the useful add-ons used in traffic prediction, and last but not least, we discuss the open challenges in the traffic prediction field.

1. Introduction

With the progress of urbanization and the popularity of automobiles, transportation problems are becoming more and more challenging: traffic flow is congested, accidents are frequent, and the traffic environment is deteriorating. The question of how to improve the capacity of the road network has attracted attention from an increasing number of scholars. To solve this problem, the first solution that occurs to most of us is to build more highways, expanding the number of lanes on the road. However, according to the study done by Dechenaux et al. [1], expanding the road capacity will cause more serious traffic conditions. One efficient way to improve the traffic environment is to establish an efficient and accurate transportation system, which can help us better arrange transportation resources, disperse the traffic flow before it is overloaded, and even provide more abundant on-road entertainment. The Intelligent Transportation System (ITS) is one of the most famous of these systems [2–4]. ITS is a complex system that is integrated by a variety of advanced technologies, e.g. transportation communication systems. Meanwhile, by taking advantage of the development of the 5G communication system, abundant on-road sensors, etc., ITS can improve traffic efficiency, ease traffic congestion, increase road capacity, and reduce traffic accidents and environmental pollution [5–8].

* Corresponding author.
E-mail addresses: boukerch@site.uottawa.ca (A. Boukerche), jwang440@uottawa.ca (J. Wang).

https://doi.org/10.1016/j.comnet.2020.107530
Received 27 May 2020; Received in revised form 15 August 2020; Accepted 31 August 2020
Available online 3 September 2020

As a vital part of ITS, an accurate and efficient road traffic prediction system can provide continuous and precise road status information based on past road conditions [9,10]. That information can be useful for the applications involved in ITS, such as traffic congestion control, traffic light control, vehicular cloud (VC), etc. [11]. Here, we take VC as an example. One difficulty in implementing and maintaining a VC lies in computing the available redundant vehicular resources on a given road segment, so as to better determine the possible work load of the cloud. However, the on-road sources are mainly gathered from the high mobility vehicles on the highway or on urban roads, which makes it so important for the cloud system to determine how many vehicles will be on the given road segment in the future. To deal with this, the traffic prediction system will provide highly reliable future traffic volumes to the VC according to the historical traffic pattern and the spatial relation over the whole road network. That will give the VC opportunity to simulate the future work load and possible computing capacity [12].

As we said in the above paragraph, the traffic flow is predicted according to the historical traffic patterns. In another word, the interaction between different traffic patterns over time will finally determine the traffic volume in the future [13]. In reality, the traffic flow is affected by the variety of traffic pattern types. In general, the seasonal trends and regularities is affected by long-term traffic features. Meanwhile, the non-linear and non-seasonal features which are uncertainty occur is affected by short-term traffic pattern. For example, traffic flow will keep showing a rapid increment in the morning and rapid decrements in the evening almost every weekday. That is a long-term feature for traffic flow affected by the working time of the society. Meanwhile, the traffic flow may show uncertain fluctuations caused by a sudden weather change or traffic accident. Three main types of the variation of time-series that affects the traffic flow are widely adopted: (1) Trend variation: The variation of processes is time-dependent and presents the trend of growing, declining or stable in a certain direction; (2) periodic change (seasonal change): The processes present the cyclic changes /(varies) at regular intervals; (3) random variation: The processes vary by external factors/influences and show disordered variations [14]. Traffic flow is generally a combination of the above variations. Because road traffic is affected by many natural and human factors, it has complex nonlinear characteristics, which creates difficulty in accurately predicting traffic.

For further introduction convenience, we would like to give a clear mathematical definition about the traffic flow prediction task, as follows:

$$\mathfrak{A}(\mathrm{X}, \mathcal{F}, \mathrm{G}) \rightarrow \hat{\mathrm{X}}, \tag{1}$$

where $\mathfrak{A}$ is the prediction model we choose for the task, $\mathrm{X} \in \mathbb{R}^{C_i * M_i * N_i}$ is the input data, $\mathcal{F}$ is the context features related to the task, e.g., the weather condition, the social media information, etc. G is the spatial feature related to the location where the prediction is made. $\hat{\mathrm{X}} \in \mathbb{R}^{C_r * M_r * N_r}$ is the prediction results. To be more clear, C_i, C_r is the number of the channel of the input and output, M_i, M_r is the number of spatial points in the input and output, and N_i, N_r is the length of the time-steps in the input and output. Furthermore, we can split the task into smaller parts according to its input and output as follows, based on past researches:

- According to the type of X and $\hat{\mathrm{X}}$: Traffic flow prediction, traffic speed prediction, etc. For example, in [15], the authors took the traffic flow dataset collected from the freeway in the Greater Tokyo Area as the input to the prediction model. The output of the model was also traffic flow. While in [16], the model used traffic speed information collected in Liuliqiao District, Beijing, to predict the value traffic speed on each road segment in the next timestamp.
- According to the dimension of X and $\hat{\mathrm{X}}$:
  - If $C_r > 1$: Multi-task prediction task. For example, in [17], the authors predicted traffic flow and speed at the same time by adding a multi-task layer at the end of a Deep Belief Network.
  - If $N_r > 1$: Multi-step prediction task. For instance, in [18], the authors predicted the traffic flow in the next six timestamps at the same time by using the iteration method and the auto-correlation coefficient method.
- According to the time interval between X and $\hat{\mathrm{X}}$: Short-term prediction task, long-term prediction. For example, in [19], the prediction intervals were from 30-min to 120-min, which can be considered a long-term prediction. While in [20], the prediction interval was set to be 5-min, which is a short-term prediction task.

In the past few years, a large number of papers around the world have reviewed state-of-art traffic prediction methods at their time. In [21], Nagy et al. not only reviewed prediction models classified in naive models, parametric models and non-parametric models, they also had a deep review of different types of open-source dataset. The dataset is classified by the mobility of the collector sensor, and the authors compared advantages and disadvantages of these datasets. In addition, the authors compared different types of data models used in traffic prediction tasks according to different contexts addressed by the task.

In [22], Do et al. put more focus on the neural network-based (NN) models used in traffic prediction tasks. The authors first introduced the basic concepts used in NN model prediction. The authors then reviewed the NN models categorized by their neural structure, layer structure, and activation function. In the end, a summary of the reviewed NN models is made based on different prediction situations.

Along with the development of the Deep-Learning structure in the NN model, some papers have also focused on the Deep-Learning-based (DL) model used in traffic flow prediction. In [23], the authors reviewed DL models and the related work-flows used to build up a usable DL prediction model. The authors also compared the DL models by applying them to different kinds of traffic state prediction tasks using the same dataset, which can give the reader a more intuitive view of the advantages and disadvantages between different DL prediction models.

According to the papers under this topic, we have found that Machine Learning-based (ML) is more and more popular for traffic flow prediction task. The reason is that less prior knowledge about the relationship among different traffic patterns for model building, less restriction on prediction tasks, and have better non-linear features [24]. In this paper, we will focus on the ML models used in traffic prediction tasks. Meanwhile, we also pay attention to the applicable scenarios the ML model has been applied to. It is important that we compare different types of models on their accuracy, as well as features such as the ability to deal with some specific problems, and the efficiency of the model, the hardware and data dependency of the model.

As we know, ML is a very huge topic, and there are many kinds of classification methods for ML models based on different perspectives. In this paper, the methods we reviewed are categorized by different types of ML algorithm theories, as illustrated in Fig. 1 according to [25–28], and are further clarified as follows:

- Regression model: By studying the relationship between the dependent variable and the independent variable, the regression model tries to use a curve or a line to fit the dataset.
- Example-based model: The example-based model solved the prediction task by comparing the similarity between the input sequence and the historical data samples, and uses the found samples to make the final prediction. In this paper, we mainly focus on the k-Nearest Neighbors (KNN) model.
- Kernel-based model: In the kernel-based model, we use kernel function to map the input data into a high-order vector space, where the prediction tasks are easy to solve. In this paper, we mainly focus on the Support Vector Machine (SVM) and Radial Basis Function (RBF) model.

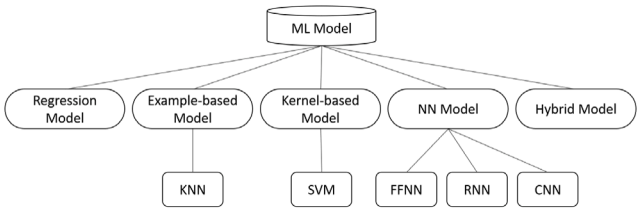

Fig. 1. ML-based models.

- NN model: NN model is a type of model built up by simulating the way information passes through neurons in the brain. The input data is passed through different network structures in different types of NN models. In the network, the input data will be transformed into an activation signal by using the activation function. In the end, the final prediction is made based on the activation signal. The basic NN models reviewed in this paper include Feed Forward Neural Network (FFNN), Recurrent Neural Network (RNN), and Convolutional Neural Network (CNN). Meanwhile, the basic NN models can be aggregated into DL models with different structures, which we will also review in this paper.
- Hybrid model: In the hybrid model, the final prediction is made by combining two or more prediction results from different prediction models.

The paper is organized as follows: In Sections 2–4, we will review the existing ML models applied in traffic prediction based on the previously mentioned taxonomy. In Section 6, we will review the open-source datasets that can be used for the traffic prediction task. In Section 7, several useful add-ons will be reviewed according to their responsibility in the process of building up a traffic prediction model. Then, we will review the existing open challenges in the traffic prediction task, as well as in the application of the ML model in Section 8. At the end of the article, there will be a short conclusion.

2. Machine learning-based models — regression models, example-based models and kernel-based models

In this section, we will review the Regression models, Example-based models and Kernel-based models used for traffic prediction tasks. Although we will give you a general introduction about each ML model type, we mainly focus on the prediction models proposed in the recent ten years. In addition, we will review different organization structures when we apply the model to a real prediction task.

2.1. Regression model

Regression models are considered as typical parametric methods. If a traffic prediction method has pre-supposition about the distribution of the traffic pattern, we can consider the method as a parametric method. Regression models are used for traffic prediction tasks because they are easily implemented and suited for traffic prediction tasks on a simple traffic network.

According to [29], in the parametric method, the mathematical model and related parameters between inputs and outputs have been determined in advance, and the relationship between each parameter and the input data is relatively certain. The parameters that are not determined in advance will be obtained by analyzing the training data. Therefore the choice of parameters and mathematical models largely influences prediction accuracy. The parametric method has been proven by many researchers to give fairly good predictions due to its reasonable mathematical and theoretical foundations.

The most famous regression model used in traffic prediction is linear regression. To use a linear regression model, the prediction results are considered to be a linear combination of existing traffic variables, and the researchers obtain reliable prediction results by selecting appropriate weight parameters and appropriate traffic variables. Many researchers use linear regression models to design traffic prediction models. The experimental results also show that linear regression models have higher efficiency and can produce satisfying prediction results.

In [30,31], Rice et al. showed the good performance of the linear regression model while giving only a small training dataset. The authors provided a prediction method that combined linear regression with time-varying coefficients. This method lets the prediction model better select the parameters in the linear regression model based on time variation. The authors claimed the model could perform better than the k-nearest-neighbor (k-NN) method, which has a window size of 20-minutes and the number of nearest neighbors to be 2. However, the authors did not give us specific prediction accuracy data. As the authors were saying in their paper, they mainly focused on the easy implementation and prediction efficiency, but not how can they get better prediction results.

The linear regression can also deal with the spatial–temporal relationship in a road network, and use this information for prediction. The authors in [32] tried to figure out the relationship between future traffic flow on a given link, and the origin link's and adjacent link's traffic flow records. It is intuitive to the authors that the upstream traffic has a strong effect to the current traffic. Therefore, the authors provided three combined prediction models, and each of these was combined with different traffic variables such as upstream traffic, current traffic, and historical average, using predetermined weighting parameters. Meanwhile, the authors also proposed a method for dynamically selecting weight parameters in a linear model based on current traffic information, and tested the robustness of the model on peak-hour traffic data.

According to the authors' experiments, although the model performed better in the 15-minute range than Historical Average (HA) and final decision rule, several drawbacks for the simple linear regression model can be addressed. First, the model has difficulty dealing with long-term traffic flow prediction task; the accuracy of the model dropped lower than HA when the prediction interval exceeded 30-min; Second, it is costly to find a set of adequate parameters to express the relationship between the traffic pattern we choose, and the found parameter can only fit for the specific road section. Finally, because the model has a time consumption problem dealing with complex road structures and more traffic patterns, the computing time for determining parameters will increase exponentially.

To solve the time consumption problem in finding the adequate parameters, Kwon et al. combined linear regression with stepwise-variable-selection method and tree-based method in [33]. The model makes a prediction using flow, occupancy and historical travel-time information for I-880 freeway. Kwon et al. used cross-validation (CV) to avoid intersection problem, and they tested the model in different prediction headway from 0- to 60-minutes. The authors found that the more recent traffic information was very helpful for short-term traffic forecasting, while historical data was good for longer-range forecasting. The use of tree-based method can highly improve the efficiency of finding the parameter by controlling the depth of the tree. However, the model's accuracy is still highly reliant on the search range we initialized, which requires in-depth understanding about the prior knowledge of the given road section.

According to the papers we have reviewed, another drawback of simple linear regression model is that the model has difficulty capturing the non-linear vibrations in traffic flow. To better predict the stochastic variation in the traffic pattern, researchers combine regression with other methods.

In order to better fit the nonlinear features in the traffic pattern, Fei et al. [34] integrate a Bayesian inference framework with a dynamic linear model (DLM) for short-term travel time prediction to better capture the non-stationary characteristics of the traffic pattern, and

adapt the Bayesian DLM into an adaptive control system to better track travel time variation . The authors observed significant improvement from the other prediction models in robustness and accuracy.

The authors of [35] demonstrated another approach to improving the model's capability to capture the non-linear features in traffic records, by providing a dual Lasso phase model with Granger causality theory to solve the traffic flow prediction. To make the prediction, the data would first be preprocessed by using a historical average method to find the residual trend of the original input sequence. Then, the first Lasso regression model was used to filter the unrelated records. The second Lasso regression, i.e., the robust Lasso regression, is used to minimize the influence from the extreme situations. As pointed out in the paper, the second Lass regression phase was solved in a two-step style by using the Alternating Direction Method of Multipliers.

To further solve the problem of the model's robustness and accuracy for short-term prediction, the authors in [15] combined latent factor model bi-linear Poisson regression model for a short-term traffic flow prediction task by using the short-term historical records from the related road segments. The base model was a bi-linear Poisson regression model. The authors added the idea of a convolutive mixture to the model in order to solve the time lag problem among different road segments. Meanwhile, the authors made the model suitable for online updating and prediction tasks by introducing a stochastic variations Bayes model to the regression model.

The regression models used for traffic prediction in the last ten years are found in Table 1. As we can see from the table, there are not many traffic prediction models based on the regression model present in recent years. The reason is that though the regression model is easy to implement, the simple structure makes the model not good at describing the continuous traffic records' temporal features. However, the regression model is still a choice for traffic prediction tasks for small and simple-structure traffic networks for less computation. Meanwhile, regression functions such as LASSO can also guarantee the model's ability to capture the non-linear features among the traffic record.

As we can see in this subsection, the regression model has had great development in a relatively long period. The adequate parameter set has great influence to the accuracy of the model. Scientists in this area have tried different methods, from stepwise-variable-selection method and tree-based method [32,33] to the methods based on least squares method. The improvement of the parameter-finding-method has highly improved the model's ability to capture short-term traffic vibration. We also notice that the authors tried to utilize the spatial connection information between each road segment in their model. According to the latest research [15], the regression can achieve high accuracy and robustness on the multiple-road-section prediction task.

However, we also notice that the regression model has its own drawback. To build up a traffic prediction model by using the regression model requires only a small historical dataset. It is good for fast implementation tasks and tasks that cannot prove abundant historical data. However, at some level it limits the accuracy potential of the model, for the model does not show a microscopic view of the dataset, such as its seasonal changes, and can easily have an overfitting problem.

2.2. Example-based model

As we indicated in Section 1, we focus on the KNN models for the example-based model. One of the main reasons for using the KNN model for traffic prediction is that it has perfect features capturing the spatial relationship between the road segment in the traffic network. Sometimes, the model has also been used as the tool to eliminate unrelated traffic data to the current prediction task. According to [40], the k-nearest-neighbor model is a data-based non-parametric regression method. Instead of establishing a mathematical prediction model, it searches for the K nearest neighbors that match the current variable values and uses that K data to predict the value at the next period. In this method, the information necessary for prediction is obtained from historical data. The historical database contains various trends and typical patterns of traffic statuses. Each type of data in the dataset represents a possible trend of traffic evolution. The algorithm considers that the relationship between all factors of the traffic system is contained in the historical dataset, therefore, the quality of the historical database, and specifically whether it stored all the traffic states that may appear in the future, is of great significance to the prediction results and accuracy.

The normal KNN model is widely used in the classification problem. However, when KNN model is applied to the regression problem, it can be modified in two steps: K-means clustering and non-parametric regression forecasting based on the clustering analysis [41]. The K-means clustering algorithm uses distance as an evaluation metric for similarity. The algorithm considers that the cluster is composed of objects that are close to each other. It requires that the objects in the same cluster have a high degree of similarity. In K-means clustering, the first step is to set the number of clusters (K). Some sample points are then randomly selected as the initial cluster center, and the rest of the data is then aggregated to the cluster center according to the principle of proximity. Then, the new center of each cluster are calculated. The clustering is repeated until the cluster center position converges. In the prediction step, the state vector, which describes the current state, selects the neighbor data, and finally the neighbor points are used to predict the value of the next moment.

Many researchers have implemented the KNN model to make traffic predictions. To use KNN model for traffic prediction, four main challenges are pointed out in [41]: (1) how to define an appropriate state vector, (2) how to define an appropriate distance metric, which is related to how similarity between data points is determined, (3) how to generate forecast, and (4) how to manage the potential neighbor database (when the database is too small, it cannot contain enough useful data to make a prediction, but if it is too large, the computation cost will be greatly increased). In this section, we will focus on the above problems, and compare each of the existing traffic prediction models using KNN theory.

The common way to use KNN for traffic prediction is to first select the K candidates from the dataset by using the Euclidean distance, then output the result by using the weighted average algorithm. In [36], the authors used this method for traffic flow prediction in different prediction intervals (15, 30, 45, 60-min). In this paper, the dataset was made by historical data collection on Friday and Saturday on the given road section. In their experiments, the authors provided a thorough study on the effect of the length of the records in the dataset, and the number of neighbors for different lengths of prediction intervals and different weekdays. Moreover, the correlation between the hyper-parameters was also studied in this paper. The authors addressed how the number of neighbors affects the accuracy of the model following a concave trend. Meanwhile, the number of k is relatively small while the effect from the length of the records is bigger.

To further improve the K candidates selection process, and to make time correlation information play a more significant role in the prediction model, the authors in [38] chose correlation coefficient distance as the distance metric to select the more related neighbors. The distance of the neighbors was not only impacted by the data records itself, but also the time interval from the current time, which will be infinite if the time interval is more than the threshold preset (no more than one day). The candidates for the prediction were selected by using the local minimal algorithm to avoid the influence from the dimensionality reduction of k because of the phenomenon overlap while using the normal selection process. To make the final prediction, the authors introduced a Linearly Sewing Principal Component method (LSPC), which leads the final prediction into a minimization problem. The model outperformed other models in the experiments on different road segments.

In addition to the temporal features in the traffic flow dataset, we should also consider the spatial correlation effect. In [39], the authors

Table 1
Regression models used for traffic prediction in the past ten years.

Papers	Tasks	Regression model	Input–Output	Area	Additions
Okawa et al. [15]	Flow	Bi-linear regression model	Multi–Multi	Urban area	Online prediction; Optimized by Stochastic variations Bayes method; capture spatial correlations use convolute
Li et al. [35]	Flow	LASSO $\lambda = 50\,000$	One–One	Freeway	Based on Granger causality theory; Two Lasso phase.

Table 2
KNN models used for traffic prediction in the past ten years.

Papers	Task	State vector	Distance metric	Prediction method	Additions
Chang et al. [36]	Flow	Target detector's historical records	Euclidean distance	Weighted average	Grid search for k and d
Zhang et al. [37]	Flow	Target detector's historical records	Euclidean distance	Weighted and non-weighted average	k = 18, d=4; Data preprocess use Min–Max Normalization and reasonable check
Zheng and Su [38]	Flow	Target detector's historical records	Correlation coefficient distance	LSPC	k = 10
Xia et al. [39]	Flow	Target road & upstream & downstream historical records	Distance consider spatial–temporal traffic	Weighted average and trend adjustments	K = 4; MapReduced KNN model;
Cai et al. [16]	Speed	Historical speed records. The grid distance information	Gaussian weighted Euclidean distance	Gaussian weighted average	K = 10; K first been reduced by equivalent distance based on grid distance.

introduced a MapReduce-based KNN model for traffic flow prediction. One big difference between this model and the above KNN is that this model considers the spatial influence. The distance metric depends on the given road segment, as well as its upstream and downstream. The authors also used Distance Weighted Voting to reduce the influence from the number of K. As for the final prediction, the authors used a weighted average method with a trend adjustment algorithm. The model was tested during a peak hour on the freeway, and the results were far better than the other models compared in the paper.

With the same purpose of utilizing the spatial–temporal features in the dataset, the authors in [16] chose another strategy. In [39], the authors mainly focused on the upstream and downstream of the given segment, and a weights vector was used to determine how much influence the other road brings to the given road segment. However, the authors in [16] first decreased the number of potential candidates by using equivalent distances influenced by the grid distance between the given road segment and the other. This spatial correlation ratio was compared to the preset threshold to filter the less spatial-related road segment. Then, the Gaussian-based Euclidean distance was calculated for finding the candidates for the final prediction. Finally, the prediction was made by using the Gaussian weighted average algorithm concerning the distance metric. The advantage of the method in this paper is that it can capture the useful information not only on the adjacent road segment, but over the whole road network.

Table 2 shows the KNN models used for traffic status prediction in the past ten years. By using the KNN model, the system can extract the interactions between road segments from the traffic database. According to different types of the state vector, the interaction can be described by geographical distance or by the distance of the continuous traffic records from different segments. Meanwhile, the elements the state vector combined with is also essential for the accuracy of the prediction results. The historical data from the target road segment and the data from its upstream and downstream are used to enrich the information in the state vector [39]. However, the extension of the state vector may increase computation, and the cost of maintaining the historical dataset. As for the distance metric, if the state vector contains a single type of records, such as the historical traffic flow records, Euclidean distance is the choice for most researchers [36,37].

Table 3
Common kernel function.

Kernel name	Kernel function
Linear	$k(x_j, x_k) = x_j' x_k$
Gaussian (RBF)	$k(x_j', x_k) = exp(-\|x_j - x_k\|^2)$
Polynomial	$k(x_j', x_k) = (1 + x_j' x_k)^q$, where q is in the set $\{2, 3, \ldots\}$

However, while there are different types of records, it is better to choose a weighted distance function to balance each type of records' influence. We also noticed that more information added into the dataset of the KNN model increases the calculation time, and in [16], authors tried to reduce the search space for the candidates' selection according to the spatial distance. Still, the calculation is much bigger. Also, we have seen spatial–temporal features beginning to be considered in the distance calculation, but the final prediction is still made for only one road segment, which will also lead to greater consumption if the model is applied to a complicated road network.

2.3. Kernel-based model

In the kernel-based model, we focus on the Support Vector Machine (SVM). SVM is a statistical learning theory for classification and regression problems which is proposed by Vapnik [42]. Based on the principle of structural risk minimization, SVM can avoid the disadvantage of being easy to fall into local optimum compared with other nonlinear prediction models, and has a strict statistical theory foundation. SVM is widely used in the classification problem; when applied to a regression problem, some people would like to call it Support Vector Regression (SVR). The main idea of SVM when dealing with regression problems is to establish a hyperplane as the decision surface for a given training sample so that all sample points are close to the hyperplane, and the total deviation of the sample points from the hyperplane is minimized.

When the traditional SVM model deals with regression problems, the hyperplane is generally generated by a linear function, while the traffic prediction belongs to the non-linear regression problem. Therefore, the researchers transformed the traffic prediction problem into a

linear regression problem in high-dimensional space by using the kernel function.

Here we replace $\Phi(x_i) \cdot \Phi(x)$ with $k(x_i, x)$ which is known as kernel function. Kernel function enables $\Phi(x_i) \cdot \Phi(x)$ works in high-dimensional space using a low-dimensional space data input, and we do not need to know the transform function Φ. The most widely-used kernel functions are shown in Table 3.

SVR was first found to be used in the traffic prediction field in [43]. SVR was used to predict the traffic flow of one given crossroad. The prediction interval was set at one hour, and the past five time steps were used to predict the next hour, which can be seen as a long-term application scenario. The training time was 8 a.m. to 4 p.m., and the data used to test was collected from 6 p.m. to 10 p.m. on the same day. The average errors were 6.03%. In [44], an SVR that used RBF as kernel function was compared with Multi-Layer Feed-forward Neural Network (MLFNN) to forecast the traffic speed aggregated in 2-minutes. The input data is the speed data of the past 10 min, and the output of the model is the speed data in the next 2-minutes, 4-minutes, and 6-minutes. From the results provided by the authors, we can see that SVM can outperform MLFNN when the training dataset is collected from the past two days. With the expansion of the training set, the Mean Absolute Percentage Error (MAPE) of SVR slightly increases, while the accuracy of NN increases and exceeds SVR, which shows that SVR has advantages when the dataset is small. By introducing a new parameter — v, authors of [45] tried to solve the optimization problem of SVR in a better way. The new SVR with additional parameters in [45] called v-SVR was used to forecast the traffic volume on the freeway. The MAPE and RMSE of the new model were at the range from 5.5% to 8.8% and 44.7% to 64.5% respectively, which were better than the results given by MLFNN.

As we said, SVM is a type of kernel-based model. An adequate chosen kernel function will greatly affect the accuracy of the model. The most famous kernel function is RBF. The authors in papers [46,47] both used RBF as their kernel function. Meanwhile, some scientists have tried to consider seasonal features in the kernel function. In [48], the authors introduced two seasonal kernel functions, i.e., seasonal RBF function and seasonal linear function, to the SVM model; these were used to help the SVM model make use of the seasonal information among the traffic records. Through the kernel function, the time interval information will also be considered with respect to the preset seasonal period. From the experimental results, the model performed better than most of the other prediction models, such as ANN, ARIMA with Kalman filter, SVM with the non-seasonal kernel. But the model did not perform better than the SARIMA with Kalman filter, except in the morning peak time. The authors also studied the training time and the prediction time for a different model, which shows that even though the accuracy of the SARIMA model is little higher than the seasonal SVM model, it costs twice as much training time than the seasonal SVM model, which makes the seasonal SVM model more competitive than SARIMA.

After choosing adequate kernel function, it is important to optimize the parameters for SVM. The difficulty for parameter optimization to avoid falling into a relatively lower local optimal, while, keeping the time consumption at a low level. The authors in [47] chose an algorithm called Continuous Ant Colony Optimization (CACO) to optimize the parameters in their SVM model. CACO was developed from the idea of ACO [49]; the difference is that in CACO, the search space is set to be successive. The SVM with a Gaussian RBF kernel function was used to predict the traffic flow during the two peak times, i.e. the morning peak time (6 am to 10 am) and the evening peak time (4 pm to 8 pm). With the help of CACO, the model can easily decide the parameters depending on different features of input data. The data was collected from urban Taiwan, and the results showed that SVM with CACO can outperform SARIMA in all situations.

Another optimization algorithm was used in [50]. The authors introduced a method called Genetic Particle Swarm Optimization (GPSO) by adding crossover and mutation operation to the original PSO algorithm. Additionally, the author introduced Empirical mode decomposition (EMD) to decompose the input sequence into several sub-frequencies at different fluctuation levels. For different sub-frequencies, SVM with different kernel functions were applied. In their experiment, the authors further proved that genetic and decomposition thought can be helpful to improve the accuracy of the SVM model.

To further improve the accuracy of the SVM, some scientists have focused on the data pre-processing progress. In [51], the authors provided a SVM model to deal with the chaotic and non-stationary features in traffic speed records. The main thought of the paper is to map the data into a high-level space, which can easily capture non-linear and chaotic characteristics. To do that, first, the raw data input into the SVM model was de-noised by decomposition and wavelet transformation. A soft-thresholding approach was used to address the threshold, and for the transformation, the authors chose the Symlets family of wavelets. After de-noising, the data was input into the SVM model after normalization. According to the Phase Space Reconstruction (PSR) theory, the data was further embedded into a higher dimension by using a G-P approach. The kernel function to the SVM model was set to be Mexican-hat wavelet function. According to the experiment results, the proposed model showed better performance than the regular SVM model in an unusual situation, indicating that the model can better address the non-recurrence time-series.

In [53], the authors chose another decomposition method, i.e., Discrete Fourier Transform (DTF) for the RBF kernel SVM to deal with the traffic flow prediction tasks on the freeway. The method was used to decompose the input sequence into a common stable frequency and the residual series, to make a better prediction on the burst situation in the traffic sequence (in this paper's experiment, the burst situation means the morning and evening peak hour). From the results proposed by the authors, the DFT shows better performance than the EMD methods.

In addition to the model's accuracy, the scientists also paid attention to the problem that occurred in the practical application. In [46], the authors provided an online-SVM model (OL-Model) using RBF kernel to deal with the prediction task in unusual traffic situations. The parameters were updated each time a new data record was added to the dataset. The size of the moving window for the OL-SVM was set to be 10, and the data resolution was set to be 5-min, containing more nonseasonal vibrations. The model was compared with the Gaussian maximum likelihood (GML) approach on both usual and unusual traffic situations in the morning peak time. According to the results, the OL-SVM did not perform as well in the usual situation as the GML approach, but it outperformed in an unusual situation.

Another practical problem is multi-step prediction. In [52], the authors tried to solve the multi-step prediction problems. The difference of the model in this paper from the original SVM is the structure of the input data. The input data can contain historical records such as the previous time interval, and can also contain the records from the upstream. According to the experiment results, while the prediction interval was set to be 1, the input with upstream records and the previous records perform better, because in the short time interval, the traffic flow from the upstream can highly influence the given road section. While the prediction interval went higher from 1 to 3, the accuracy for all models taking different kinds of inputs went down, but the model that took historical and upstream data performed better than the other models. When the interval increases, more information in the historical record becomes useful to the prediction task. In the end, the authors also pointed out that, considering the computation consumption, the input with only the previous and historical records of the given road section was the best choice for implementation.

In Table 4, we compared the SVM models used for traffic prediction tasks in the past ten years. As we can see, most of the SVM models chose RBF as its kernel function, which has great features for dealing with non-linear fluctuation. We also notice that the data preprocessing, especially the decomposition process, can greatly increase the SVM model's

Table 4
The SVM model proposed in the past ten years.

Papers	Kernel	Parameters	I/O	Area	Additions
Castro-Neto et al. [46]	RBF		One–One	Urban	Online prediction
Hong et al. [47]	RBF	Optimized by CACO	One–One	Urban	
Wang and Shi [51]	Mexican-hat wavelet function	a = 6, $\alpha = 0.0001$, C = 1000	One–One	Freeway	Embed input data by G-P approach.
Lippi et al. [48]	RBF + Linear (Season)	S = 480 S-RBF: C = 5, $\epsilon = 0.01$, $\gamma = 1$, $\gamma^S = 192$, $\omega = 3$ S-Lin: $\lambda = 0.001$, $\epsilon = 0.01$, $\omega = 3$, T = 150 000	One–One	Freeway	
Mingheng et al. [52]	RBF	Optimized by grid search	One–One	Urban	Online prediction
Duo et al. [50]	RBF, Polynomial, Linear for different frequency fluctuation	Optimized by GPSO C = 98.24, $\varepsilon = 0.021$, $\sigma = 0.68$	One–One	Urban	Decompose the original sequence by EMD
Luo et al. [20]	RBF		One–One	Freeway	Make decomposition using DTF

accuracy. In [46,52], authors chose the SVM model as the kernel of the online traffic prediction system. The reason is that SVM has a lower computation requirement, which is suitable for the quickly updated requirement in online traffic prediction tasks. However, SVM has its problem; limited by its natural structure, it has difficulty dealing with a big traffic network problem, which is one of the trends these days. Besides, the cost of finding appropriate hyper-parameters is relatively big compare to KNN or regression model while using grid search [52]. However, methods such as CACO [47] and GPSO [50] are used to quickly find the hyper-parameters' values.

3. Neural network-based models

The neural network-based (NN) models belong to the category of non-parametric models. The term Non-parametric models does not refer to a model that requires no parameters at all, but means that the parameters in the model do not need to be preset; rather, they are obtained by learning historical data [41]. These kinds of models do not require prior knowledge, but sufficient historical data, which is an advantage in the current era of information explosion. By learning historical data, the model establishes an intrinsic link between prediction aim and historical data.

According to several previous studies [41,54] the non-parametric method can better catch the non-linear feature of the traffic pattern compared to traditional linear regression model, which highly increases the precision of the algorithm. But the non-parametric method also has its disadvantages [55]. The model has high requirements for the quantity and quality of historical data, and the cost of training a non-parametric model can be large compared to other kinds of methods.

NN model was first proposed in the 1940s by Warren S. McCulloch and Waiter Pitts [56]. NN model is an abstract mathematical model that can better identify the complex linear system. Meanwhile, NN model uses the black-box learning model, which is very suitable for traffic prediction. It does not require any empirical formulas to obtain the inherent relationship of the data from the dataset. By learning a large number of input and output samples, the NN model automatically adjusts and establishes the input and output map model. The NN model has the features of associative memory, strong fault tolerance, and robustness, so it is widely used in traffic prediction. Moreover, the NN model can update the network according to real-time traffic information, which can ensure real-time prediction. The traffic system is influenced by many factors, which can be captured by ANN. It uses the historical data of the research road section as well as the relevant road section and various factors affecting the transportation system, such as weather, road construction, accident, road conditions, etc.

However, a large amount of data is needed in the training process of the neural network, and insufficient data may lead to poor prediction results. The trained network is only suitable for the current road section, and when the road conditions and traffic conditions change, the network will no longer be applicable, so the promotion ability of the NN model is poor. Another problem is that the number of neurons in each hidden layer needs to be determined by experience. Too many neurons of each hidden layer will result in a huge network structure and long calculation time. However, if the number of neurons of each hidden layer is too small, it is difficult to ensure the accuracy of the prediction results.

In this section, three different types of NN model will be discussed: Feed-Forward NN (FFNN) model, Recurrent NN (RNN) model, and Convolutional NN (CNN) model.

Table 5
Common activation function.

Name	Activation function
Logistic	$\sigma(x) = \frac{1}{1+exp(-x)}$
Tanh	$tanh(x) = \frac{exp(x)-exp(-x)}{exp(x)+exp(-x)}$
ReLU	ReLU(x) = max(0, x)

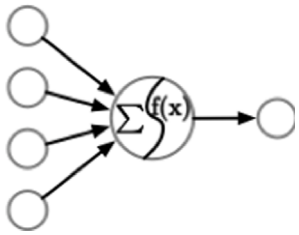

Fig. 2. Typical structure of a neuron in neural network.

3.1. Feed Forward Neural Network (FFNN)

Artificial neurons are the basic units of neural networks; a typical structure of a neuron is shown in Fig. 2. If we assume that a neuron takes d inputs $x_1, x_2, \ldots, x_d$, and the input vector to the neuron can be defined as $\mathbf{x} = [x_1, x_2, \ldots, x_d]$. The output of the neuron will be:

$$a = f(\mathbf{w}^T\mathbf{x} + b) \tag{2}$$

where $\mathbf{w} = [w_1, w_2, \ldots, w_d]$ is a d-dimensional weight vector, b is the bias variable to the neuron, and $f(\cdot)$ is the activation functions, which can enhance the expressive ability and learning ability of the network. The most commonly used activation function are listed in Table 5.

Each neuron in the FFNN is divided into different groups, according to the order of receiving information. Each group can be be seen as

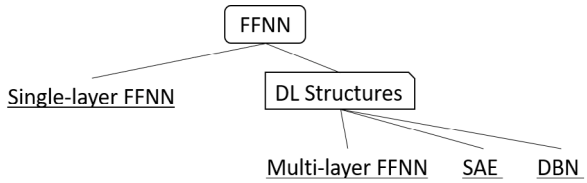

Fig. 3. Taxonomy of the analyzed state-of-the-art literature related to FFNN.

a neural layer. Each layer takes the previous layer's output as input and output to the next layer. The information in the entire network is transmitted in one direction, and there is no reverse information transmission, which is different from the recurrent neural network. The calculation inside of the FFNN can be seen in [26].

In this section, we will introduce several traffic prediction systems using FFNN. Meanwhile, we will also review several deep-learning structures for FFNN, which can highly improve its accuracy. The detailed taxonomy of the analyzed state-of-the-art literature related to FFNN is shown in Fig. 3.

The simplest and most user-friendly structure for FFNN is one-hidden layer structure. In [57], the authors used a one-hidden layer FFNN for traffic flow prediction in a single layer. Multi-Task Learning (MTL) layer was added to the end of the shallow FFNN for a multi-interval prediction. The prediction targets were the traffic flow at future time T and the one-time interval before and after. Meanwhile, the prediction for time T is the main task; the other two tasks were used to improve the accuracy of the main task by utilizing the correlations between different time intervals. As for the optimization method, the authors chose the Levenberg–Marquardt algorithm to make the NN converge faster and more accurately. In the experiment, the authors compared the MTL FFNN with the single task FFNN, which showed that MTL FFNN has better performance than single-task FFNN on different road segments.

One-hidden-layer FFNN's capability for both long-term and short-term traffic prediction is further studied in [58]. The traffic volume records of the last time step, the weekday, and the month are input to NN. The authors claimed to have encouraging results, but there are a few drawbacks on the experiments: the first is that the training data of the model is only 20% of the whole year, but the model was used to predict the traffic flow in the rest of the years. From the analysis of the dataset, we can see there are huge differences in the data from different months, particularly the total amount of the volume. From the experiment results, we notice that the model has difficulty accommodating the change in the volume; even the month information has already been put into the input sequence. The second is that from the results, we can see the model has a problem fitting an atypical situation in which the traffic flow reaches to the top of the day; this could be because simple structure FFNN cannot capture all the non-linear features among the dataset.

One way to improve the accuracy of the prediction model is to expand the dimension of the input, i.e. adding more types of traffic records. In [59], there was a shallow FFNN used for traffic flow prediction on a road containing different types of vehicles. In this paper, beside the volume of different types of vehicles, the weekday and time of the day information, the speed records of different vehicles were also taken into consideration. From the experiment, the Sigmoid activation function performs better than the tanh function. Meanwhile, Levenberg optimization works better than the Momentum optimizer. Also, with more hidden neurons, the model will be more accurate. The author further analyzed the influence of the speed records from the dataset. The results showed that the accuracy of the model increased with more information.

Until now, we only reviewed the FFNN in a one-hidden-layer structure, which is very easy to implement. However, the higher needs in terms of model accuracy requires more complicated network structure. In the rest part of this section, we will review some famous Deep-Learning structures for FFNN.

3.1.1. Deep-Learning structure for FFNN

The simplest Deep-Learning structure for FFNN is the multi-hidden-layer structure, which just adds more hidden layers into the network. In [60], the authors used a multi-layer FFNN for traffic speed prediction by utilizing the spatial-correlated detectors' records. A l_1 trend filter was used in preprocessing to denoise the original input trend, and a LASSO regression was used for choosing the spatial-correlated road segment. The model was used for both prediction tasks in normal situations and on special days with more atypical vibrations during the day. The model can perfectly handle these two situations while maintaining a stable prediction accuracy. The filters used in the model also proved useful for dealing with the spatial–temporal correlations.

However, when the network structure became deeper, the efficiency of the model was reduced. To fix this problem, the authors in [61] trained the Multi-layer FFNN in an unsupervised way by using the method called Wasserstein Generative Adversarial Network (WGAN) [62]. The model was trained for prediction in both typical and atypical situations. The input data was detrended by using the historical average method before feeding to the NN. As shown in the results, the model can provide an acceptable prediction result in either typical or atypical situations, but not better than LSTM in the usual case, or SVR and ARMA in the unusual case. As pointed out by the authors, the value of the model is that it may not be the best in only one situation, but it can fit both situations without using only one method. Meanwhile, the training time for the proposed model is much less than the other DL models, considering the number of its parameters.

In addition to multi-hidden-layer structure, there are Stacked AutoEncoder (SAE) and Deep Belief Network (DBN) which are also deep-learning structures for FFNN.

SAE is one of the deep learning structures of FFNN, used widely in traffic prediction. It is combined with multiple autoencoder (AE) layers. The basic structure of SAE is shown in Fig. 4. The purpose of each AE layer is to extract high-order features, and the dimension of the input data is reduced layer by layer. In the end, the high-order features are used to make a prediction. Using SAE can help us to extract the most useful features in training data, and reduce the computation cost [63].

According to [63], AE consists of three layers: an input layer, a hidden layer, and output layer. The process of passing input to the hidden layer can be seen as an encoding process, in which the features of the input vector are extracted. Then, from the hidden layer to the output layer, AE tries to decode those features and to reproduce the input vector. AE is trained in an unsupervised way; the idea is to minimize the difference between input vector X and output vector Z. This difference shows how well AE extracts useful features. If we assume the input vector is $X = [x_1, x_2, \dots, x_N]$, and the output of hidden layer is Y, in AE we have:

$$\begin{aligned} Y &= f(W_e X + b_e), \\ Z &= f(W_d Y + b_d), \end{aligned} \tag{3}$$

then, we can optimize the AE by minimizing the cost function defined as follows:

$$L(X, Z) = \frac{1}{2} \sum_{i=1}^{N} \|x_i - z_i\|^2, \tag{4}$$

The AE model used in [63] is given additional sparsity constraints to restrain part of the neuron in AE, which can help us to extract useful features when there are more neurons in the hidden layer than the input layer. To do that, we have to define a sparsity parameter ρ which is a number close to zero, and we define the average activation value $\widehat{rho}_j$ over the training set for the jth neuron in the hidden layer as follows:

$$\hat{\rho}_j = \frac{1}{N_T} \sum_{i=1}^{N_T} y_j(X^{(i)}), \tag{5}$$

where N_T is the number of input data over the training set. Kullback–Leibler (KL) divergence is added to the cost function, which can be

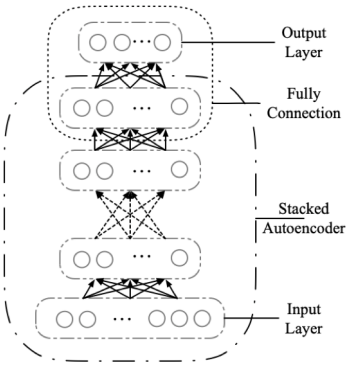

Fig. 4. Structure of stacked autoencoders.

calculated as follows:

$$KL(\rho \parallel \hat{\rho}_j) = \rho log\frac{\rho}{\hat{\rho}_j} + (1-\rho) log\frac{1-\rho}{1-\hat{\rho}_j}, \tag{6}$$

KL divergence has the feature when ρ is equal to $\hat{\rho}_j$, $KL(\rho \parallel \hat{\rho}_j) = 0$. Then we can rewrite the cost function (4) to:

$$L(X, Z) = \frac{1}{2}\sum_{i=1}^{N} \|x_i - z_i\|^2 + \gamma \sum_{j=1}^{H_D} KL(\rho \parallel \hat{\rho}_j), \tag{7}$$

where γ is the weights parameter for KL divergence, and H_D is the number of neurons in the hidden layer.

To train the SAE model, the model is first pre-trained in a bottom-up way. The first AE layer takes the input vector as input; after the first AE layer is well trained, the output from the hidden layer in the first AE layer is taken as the input to the next AE layer. After all the AE layers are trained, we have to initialize a weight matrix and a bias vector for the fully connected layer, which is also the prediction layer, then fine-tune the whole network in a top-down way using the BP method.

A comparison between the SAE model and other prediction models (SVM, FFNN, RBFNN) was made in [63]. These models were tested by making 15, 30, 45, and 60 min interval traffic flow predictions, and the results showed that SAE outperformed other prediction models with Mean Relative Error at around 6.50%. The author also claimed that the accuracy of SAE changes little, while the prediction interval is lengthened.

Another Deep-Learning structure for FFNN is DBN. Similar to SAE, DBN can be seen as a stack of Restricted Boltzmann Machines (RBM). RBM is a famous energy-based undirected graph model [64,65]. There are visible units and hidden units in the invisible layer and hidden layer respectively, and there is no connection between each variable in the same layer, similar to a fully connected network. If we define the visible units as $\mathbf{v} = [v_1, \ldots, v_m]^T$ and hidden units as $\mathbf{h} = [h_1, \ldots, h_n]^T$, the energy function of RBM can be defined as:

$$\begin{aligned} E(v, h) &= -\sum_{i=1}^{m} a_i v_i - \sum_{j=1}^{n} b_j h_j - \sum_{i,j} v_i w_{ij} h_j, \\ &= -a^T v - b^T h - v^T W h, \end{aligned} \tag{8}$$

Where $W \in \mathbb{R}^{m \times n}$, is the weight matrix for the connection between visible units and hidden units, a and b is the bias vector for the visible units and hidden units, respectively. The conditional probability for visible units $p(v_i = 1|h)$ and hidden units $p(h_j = 1|v)$ are:

$$\begin{aligned} p(v_i = 1|h) &= \sigma(a_i + \sum_j w_{ij} h_j), \\ p(h_j = 1|v) &= \sigma(b_j + \sum_i w_{ij} v_i), \end{aligned} \tag{9}$$

The free energy of RBM can be calculated as:

$$\mathcal{F}(v) = -a^T - \sum_j log(1 + e^{(b_j + \sum_i w_{ij} v)}), \tag{10}$$

and the weights and bias can be updated by calculating:

$$\begin{aligned} -\frac{\partial \log p(v)}{\partial w_{ij}} &= E_v[p(h_i|v)] - v_i \cdot \sigma(W_i \cdot v), \\ -\frac{\partial \log p(v)}{\partial b_j} &= E_v[p(h_j|v)] - \sigma(W_i \cdot v), \\ -\frac{\partial \log p(v)}{\partial a_i} &= E_v[p(h_i|h)] - v_i, \end{aligned} \tag{11}$$

The units are updated by using Markov chain as follows:

$$\begin{aligned} h^{(n+1)} &\sim \sigma(W^T \cdot v^{(n)} + b), \\ v^{(n+1)} &\sim \sigma(W \cdot h^{(n+1)} + a), \end{aligned} \tag{12}$$

And the structure of DBN is quite similar to SAE. The next layer takes the hidden units in the previous RBM as input. The way to train a DBN is the same as SAE.

Huang et al. introduced DBN used for traffic flow prediction in [17]. DBN was tested with different prediction intervals, which all yielded pretty good results. The accuracy of the model is more than 87% higher than ARIMA, SVR and FFNN based on the experiments shown in [17]. In [66], the authors also used a DBN for traffic flow prediction; the difference is that the authors adopted FireFly Algorithm (FFA) [67] to optimize the size of the hidden neurons in each hidden layer and the learning rate for the fine-tuning process. Meanwhile, according to the results, the DBN yielded the best prediction result and also kept the lowest computation cost.

In [68], a parallel training schema had been applied to DBN to reduce the training time consumption. The dataset was split into several small parts, and different DBNs with the same NN structure will update the weights and bias at the same time; then, the main DBN will integrate the weights and biases from the other DBN model, and finally the updated weights and biases will be broadcast to the other DBN model from the main model. The steps will be repeated until the loss of the model is acceptable. According to the experiment results, the accuracy of the model is almost the same as the DBN trained in a normal way, but the training time is only 25% of the normal training process.

The FFNN-based models used for traffic prediction in the last ten years are listed in Table 6. According to the existing research, the FFNN with Deep-Learning structure, such as SAE and DBN, can provide us a reliable prediction result. Meanwhile, the efficiency problem was also considered by some scientists [61]. However, as we can see from Table 6, the model is not a very popular choice for multiple road section prediction tasks. In the meantime, the model does not have a clear process to extract the temporal features in the traffic dataset.

3.2. Recurrent Neural Network (RNN)

In the feed-forward neural network, the information is passed forward, and there is no connection between neurons in the same layer. The output in the feed-forward neural network only relies on the current input. But in traffic prediction tasks, the prediction output is related to the current state as well as the previous state. To overcome this problem, RNN was introduced. RNN can better describe the influences from a consecutive traffic records [69].

The idea of RNN was proposed in the 1980s by M.I. Jordan et al. [70–72]. In RNN, neurons can accept the output from the neurons in the previous layer in addition to the neurons in the same layer, which gives the network a so-called "short-term memory". Compared to FFNN, RNN can better catch the temporal relationships among the traffic data. The detailed taxonomy of the analyzed state-of-the-art literature related to RNN is shown in Fig. 5.

An RNN can be unfolded into several repeat modules based on time, as we can see in Fig. 6. Inside a standard RNN repeat module, there is only one gate. If we assume the input time sequence is $X = [x_1, x_2, \ldots, x_{t-1}]$, where x_i is the data observed at time i, And the output

Table 6
FFNN models used for traffic prediction in the past 10 years.

Papers	Task	I/O	NN structure	Optimization	Additions
Jin and Sun [57]	Flow	One–One	One hidden layer FFNN, with Multi-task layer	Levenberg-Marquardt	The multi-task layer is for Multi-interval prediction
Çetiner et al. [58]	Flow	One–One	One hidden layer FFNN		
Kumar et al. [59]	Flow	One–One	One hidden layer FFNN (6-neurones)	Levenberg-Marquardt	Sensitive analysis
Lv et al. [63]	Flow	One–One	SAE		
Polson and Sokolov [60]	Speed	Multi-One	Multi-layer FFNN	SGD	l_1 trend filter for data preprocessing
Lin et al. [61]	Flow	One–One	Five FFNN layer (activator: Leaky ReLU; 128 hidden units)	Adam	Detrending for data preprocess; WGAN for unsupervised training
Goudarzi et al. [66]	Flow	One–One	Three layer DBN	Gradient descent	FFA used for hyper-parameters optimization
Zhao et al. [68]	Flow	One–One	Two layer DBN (100 units each)		Parallel pre-training and fine-tune

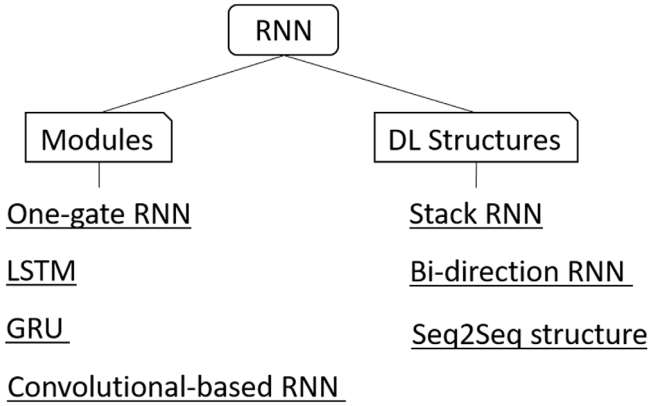

Fig. 5. Taxonomy of the analyzed state-of-the-art literature related to RNN.

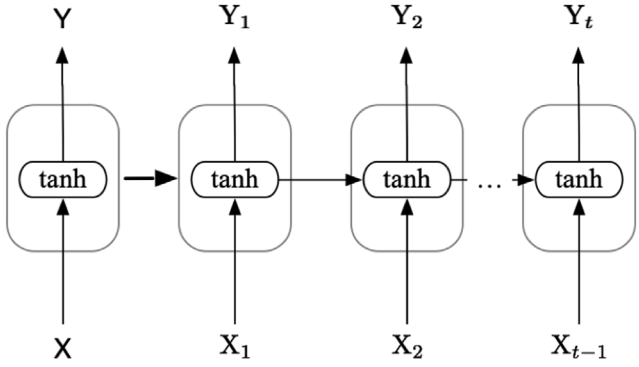

Fig. 6. Structure of a basic RNN and unfolded by time.

vector is $H = [h_1, h_2, \ldots, h_{t-1}]$, and h_i is the output based on the ith input and the output in previous timestamp, which can be shown as:

$$h_i = F(W \cdot [h_{i-1}, x_i] + b), \tag{13}$$

where F is the activate function, and the weight W and bias b is reused in every time steps, which is one of the main features of the RNN module that can highly reduce the complication of the prediction model. As we can see, the current repeat module takes the previous modules' outputs and the current timestamp input data as its input set. According to the output given by the gate structure lying in the module, the module decides how much information in the previous timestamps

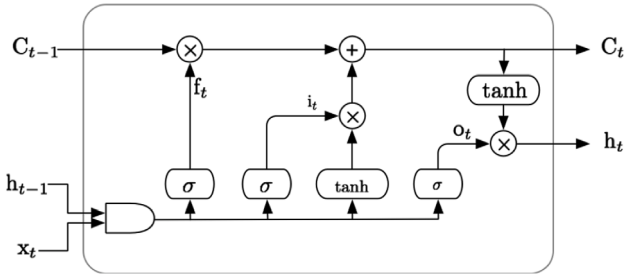

Fig. 7. Structure of the repeating module in LSTM.

could affect the current prediction result, and how much information should be transited to the next timestamp.

The training method for an RNN also follows a back-propagation way. During the optimization process, if the input series is too long, there might be a gradient explosion or gradient disappearance problem [73]. To avoid this problem, people improve the RNN model by adding gate control. The idea of gate control is to introduce the gating mechanism to control the transmission of information inside each repeating module.

In 1997, Hochreiter et al. [74] proposed Long-Short Term Memory (LSTM) module to solve the Long-Term dependency problem that occurred in basic RNN. Compared to basic RNN, LSTM has two gates inside of one at first, the input gate and the output gate. As pointed out by Gers et al. [75], the two-gates LSTM has limited when the input time series lengthened. The two-gates LSTM tends to model a linear process, and the non-linear aspect in the original dataset will be left behind. To solve this problem, Gers et al. added another gate, called forget gate, into the two-gates LSTM. The structure of the three-gates LSTM module is shown in Fig. 7. We assume that C_j is the cell state at time j, and there are three gates: forget gate f_j, input gate i_j and output gate o_j.

When using LSTM, we must first calculate the gate output using h_{j-1} and x_j.

$$\begin{aligned} f_j &= \sigma(W_f \cdot [h_{j-1}, x_j] + b_f), \\ i_j &= \sigma(W_i \cdot [h_{j-1}, x_j] + b_i), \\ o_j &= \sigma(W_o \cdot [h_{j-1}, x_j] + b_o), \end{aligned} \tag{14}$$

The gate will give us a number between 0 and 1, and we will use the output of the gate to decide how much information we want to keep. The next step is to calculate the cell state C_t:

$$\begin{aligned} \widetilde{C_j} &= tanh(W_c \cdot [h_{j-1}, x_j] + b_c), \\ C_j &= f_j * C_{j-1} + i_j * \widetilde{C_j}, \end{aligned} \tag{15}$$

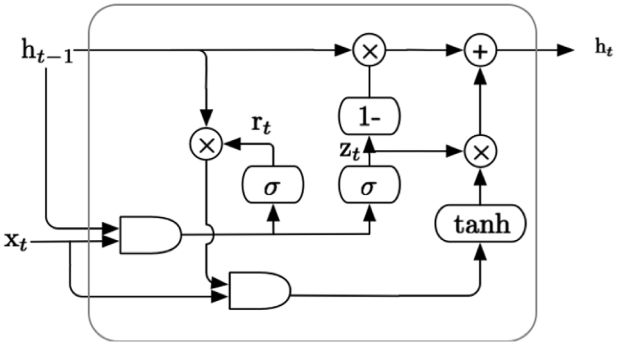

Fig. 8. Structure of the repeating module in GRU.

And the output of the module at time t is:

$$h_j = o_j * tanh(C_j), \tag{16}$$

As we can see from the above functions and Fig. 7, the LSTM module takes the current time input data and the hidden state from the previous timestamp as its input. In [76,77], Gers et al. proposed a new type of LSTM called peephole LSTM, which has also been widely used in traffic prediction tasks. In addition to the two elements in the input of the original LSTM, peephole LSTM also takes the cell state from the previous timestamp as its input to calculate the value of the forget gate and input gate. It then takes the current cell state into its input set to calculate its output gate value, as shown in the following functions:

$$\begin{aligned} f_j &= \sigma(W_f \cdot [C_{j-1}, h_{j-1}, x_j] + b_f), \\ i_j &= \sigma(W_i \cdot [C_{j-1}, h_{j-1}, x_j] + b_i), \\ o_j &= \sigma(W_o \cdot [C_j, h_{j-1}, x_j] + b_o). \end{aligned} \tag{17}$$

In [78], the authors provided a thorough analysis the performance of LSTM, and how hyper-parameters will affect the performance of the model. In their paper, the authors used an LSTM NN model with one hidden layer for a short-term traffic flow prediction task. The authors set up the hyper-parameters, i.e., the length of the input traffic flow record and the number of hidden units inside the hidden layer, for the prediction model by using the grid search method. The proposed model was compared with Random Walk (RW), SVM, FFNN, and SAE by using MAPE and RMSE. The prediction was made for the 30 detectors one at a time, and the results showed that the LSTM NN model outperformed the other models on every length of the prediction interval. The authors also analyzed how hyper-parameters affected the prediction results. According to their results, for the number of hidden units, from 5 to 20 the accuracy of the model increased, but after 20, the accuracy decreased. As for the length of the input data, the accuracy did not show many changes while the length increased for the LSTM NN model, but for the other model, aside from RM, the accuracy increased. The authors explained that this was because LSTM NN model can dynamically optimize the useful length of the input data. One drawback of the experiment setting is that the training dataset from the first 200 workdays in 2014 mainly covered three different seasons, while the test dataset was from winter. The traffic state is highly influenced by the seasonal changes, which makes the experiment setting less firm.

Another variation of the LSTM module, which is also widely used in traffic prediction task, is Gated Recurrent Unit (GRU). GRU was first proposed by Cho et al. in 2014 [79]. The structure of the GRU module is shown in Fig. 8. As we can see, there are three gates inside an LSTM module, but there are only two gates inside a GRU module, i.e., reset gate and update gate, which can be calculated as follows:

$$\begin{aligned} r_i &= \sigma(W_r \cdot [h_{i-1}, x_i] + b_r), \\ z_i &= \sigma(W_z \cdot [h_{i-1}, x_i] + b_z), \end{aligned} \tag{18}$$

and the output h_i of the module are calculated as follows:

$$\begin{aligned} \widetilde{h}_i &= tanh(W_h \cdot [r_i * h_{i-1}, x_i] + b_h), \\ h_i &= (1 - z_i) * h_{i-1} + z_t * \widetilde{h}_i. \end{aligned} \tag{19}$$

The forget gate and input gate in LSTM are aggregated into one gate — the update gate, and there is no cell state in GRU. These two main changes reduces the parameters required by the model and gives it a simpler calculation structure, which means the GRU module can be trained to predict in a faster way compared to LSTM.

The application of GRU in traffic prediction can be found in [18]. The authors applied a GRU model for a multi-step short-term traffic flow prediction task both in an urban area and rural area. Both of the data from urban and rural areas are collected from May 31 to June 7, and the data from 0 am to 5:59 am were not included. As we can see from the analysis of the two datasets, the urban data has one more huge vibration in the morning peak time, and the traffic volume drops very quickly after the morning peak. Min–Max Normalization method was chosen for the preprocessing stage. The model was compared with RBF by using MSE and MAE. The results of the experiment showed that the MSE and MAE of GRU were twice as small than RBF both in urban areas and rural. And with the help of the auto-iteration algorithm for the multi-step prediction task, the accuracy of the model did not show much change in the rural area, but the accuracy of the model decreased more while in an urban area, but still better than the RBF model or the traditional iteration algorithm.

LSTM and GRU are famous types of RNN modules. The author in [80] compared GRU and LSTM by using the traffic volume data collected by 30 random sensors from the PeMS dataset. These two models were used to predict the highway traffic flow for the next 5 min by using data from the previous 30 min. The models were trained and tested on each sensor. As shown in the proposed experiment result, the RNN structure, either LSTM or GRU, can perform better than ARIMA. Even though GRU can slightly outperform LSTM, the overall difference between LSTM and GRU is not huge. Through the histogram of the distribution of MAEs and MSE, GRU performs with more stability than LSTM, and the difference of error rate on the different sensors is smaller than LSTM.

While the RNN module and its variations above are good at dealing with time-series-type problems, it has a problem capturing the spatial features among the detectors inside the dataset. To capture both spatial and temporal correlation in the dataset, Shi et al. provided a Convolutional LSTM Network (Conv-LSTM) based on the structure of peephole LSTM in [81]. Inside of using multiplication, Shi et al. used convolution in each gate. The gates value, hidden states and cell states can be calculated as follows:

$$\begin{aligned} f_j &= \sigma(W_{fc} \odot C_{j-1} + W_{fh} \cdot h_{j-1} + W_{fx} \cdot x_j + b_f), \\ i_j &= \sigma(W_{ic} \odot C_{j-1} + W_{ih} \cdot h_{j-1} + W_{ix} \cdot x_j + b_i), \\ \widetilde{C_j} &= tanh(W_c \cdot [h_{j-1} + x_j] + b_c), \\ C_j &= f_j \odot C_{j-1} + i_j \odot \widetilde{C_j}, \\ o_j &= \sigma(W_{oc} \odot C_j + W_{oh} \cdot h_{j-1} + W_{ox} \cdot x_j + b_o), \\ h_j &= o_j \odot tanh(C_j), \end{aligned} \tag{20}$$

where $\odot$ represents the convolution operation.

Up to now, we have reviewed the some of the most famous RNN modules used for traffic prediction tasks, and their performance in a single layer structure. However, RNN also has its Deep-Learning structure. In the rest of the section, we will review several widely used Deep-Learning structures for RNN.

3.2.1. Deep-learning structure for RNN

To stimulate the RNN's greater potential, Pascanu et al. [82] and Hihi et al. [83] brought RNN into a Deep-Learning structure. As was pointed out in [84], applying deep structure to the RNN module can highly improve the accuracy of the prediction model. A deep structure

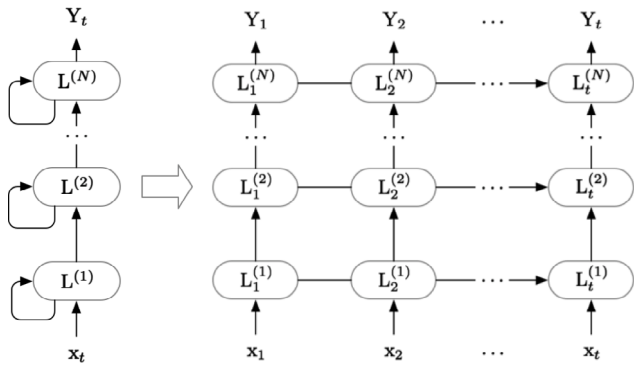

Fig. 9. SRNN structure and SRNN unfolded based on time series.

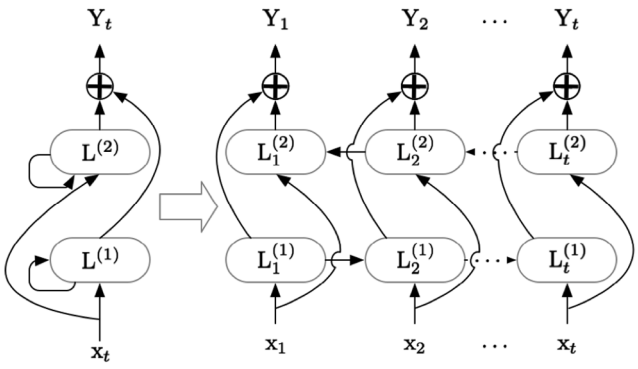

Fig. 10. Bi-RNN structure and Bi-RNN unfolded based on time series.

can capture more abstract features among the dataset, which is a benefit for the final prediction task.

One Deep-Learning structure of RNN is called stacked RNN (SRNN), the structure of which is shown in Fig. 9. In SRNN, two or more RNN layers are stacked together, and the information is passed in the same direction in each layer. As shown in Fig. 9, L_i^j is the RNN module in layers j at timestamp i, N is the number of layer in SRNN. If we assume $h_i^{(j)}$ as the output of the j layer at time i, $h_i^{(j)}$ is calculated as follows:

$$h_i^{(j)} = f(U^{(j)} h_{(i-1)}^{j} + W^{(j)} h_i^{(j-1)} + b^{(j)}), \tag{21}$$

where $U^{(j)}$ and $W^{(j)}$ are weight matrices, and $b^{(j)}$ is bias vector, and $h_i^{(0)}$ is x_i.

The performance of SRNN is studied in [85]. The authors used an SRNN DL structure to integrate three-layers LSTM. The proposed model was used for a long-term prediction traffic flow prediction in the urban area of Beijing. The input data was aggregated based on the geographical information while the area was split equally by straight lines (8*16), and the predictions were made one at a time on each grid. Meanwhile, the prediction interval was set at 1 h. To avoid the computing delay caused by redundant layers in the structure, the authors chose a three-layer LSTM SRNN as the prediction model. The authors used a 24-step time sequence as the input data. The experiment results showed that the model performed well. In the meantime, the authors studied the influence of the learning rate according to the distribution of RMSE of the prediction result, while using different learning rates. The authors found that a smaller learning rate could better help the model converge to a higher accuracy status.

However, a complex network structure will easily lead to an over-fitting situation. To avoid this, the authors in [86] used a two-layer-structure SRNN to integrate LSTM with one dropout layer at the end of the SRNN. The model was used to predict the short-term traffic speed in two areas. One of the areas had a high conjunction rate and the other one had a low conjunction rate. The dataset was aggregated into a 5-min interval. In their experiment, they focused on the influence while changing the length of the input sequence, and the dropout rate. First, from the experiment results, the use of the dropout layer can help reduce the gap between the training accuracy and the validation accuracy, which means the method can help the model avoid over-fitting problems. According to the results, while using data from the high conjunction area, lower dropout rates perform better; otherwise, higher dropout rates perform better. Meanwhile, the model using high conjunction area data is more sensitive to the changes in the dropout rate. The authors also pointed out that longer input sequences can perform better.

Another Deep-Learning structure of RNN was proposed in [87] called Bidirectional Recurrent Neural Network (BRNN), as shown in Fig. 10. In each of its layers, it is combined with two RNN layers, and the information is passed in a different direction in each layer, which is the primary difference between SRNN. Assuming Y_i^j is the output of the BRNN at time i for layer j, which can be calculated as follows:

$$\begin{aligned} h_i^{(j_1)} &= f(U^{(1)} h_{(i-1)}^{1} + W^{(1)} x_i + b^{(1)}), \\ h_i^{(j_2)} &= f(U^{(1)} h_{(i+1)}^{1} + W^{(2)} x_i + b^{(2)}), \\ Y_i &= Concatenate(h_i^{(j_1)}, h_i^{(j_2)}). \end{aligned} \tag{22}$$

Because of the structure of BRNN, this model can make predictions reliant on the previous time step as well as the time step behind. In [88], a comparison among single-layer RNN, multi-layer RNN, and BRNN was made. Meanwhile, the authors provided a deep-learning neural network combined with stacked BRNN, multi-layer RNN and Masking layer. In addition to model's accuracy, the author also focused on its scalability and the robustness. The authors also pointed out that adding more layers in a deep learning neural network would not always give us a better result.

Let us take a step further to look at the application of BRNN in traffic prediction task. In [89], the authors solved the traffic flow prediction task by using a stacked BRNN with an additional pooling layer before the prediction layer. In the paper, the authors provided a multi-layer BRNN LSTM (DBL), in which all of the BRNN layers were aggregated in an SRNN way. The data input to the model will first be passed to an embedding layer, then to the multi-layer BRNN. Then, the model will collect all the output data in each time-step. All the output data will be passed through a mean pooling layer to calculate the mean value over the output in all time-steps. In the end, the mean output is passed to a linear regression layer to make the final prediction. In the experiment, we can see that the deep hierarchy structure of BRNN can improve the prediction accuracy compared to a normal BRNN. The authors addressed that the NN should be deep enough to extract useful high-level features in the dataset. Meanwhile, the authors also tested the scalability of the model by predicting traffic flow in different lengths of time intervals (15-min, 20-min, 25-min, and 30-min), the results show that the DBL performs the best in all situations.

Beside using BRNN as the core of the prediction model, a BRNN can also be used to provide periodic features to other parts of the prediction model. In [90], the data from different detectors was aggregated into one temporal–spatial matrix. The historical data in the input set contained the data in previous T time-steps, as well as the T time-steps from one day before and one week before. The model in this paper can be divided into two phases; the first is the feature extraction stage, which is combined with one Conv-LSTM model and a BRNN model. In the Conv-LSTM model, the input in each time-step to the SRNN LSTM model will first be processed by a one-dimensional CNN and a pooling layer, which is used to extract the spatial features among different detectors. As for the LSTM BRNN, there is only one BRNN layer, which is combined with an LSTM layer of two different directions. The BRNN layer in this paper was used to extract the temporal features. Then, the second phase is the prediction phase. The spatial and temporal features

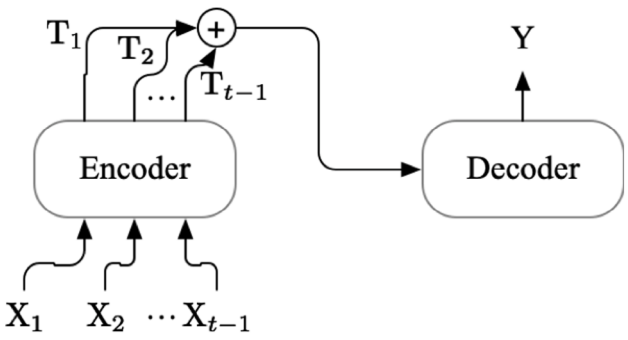

Fig. 11. Structure of Seq2seq structure.

from the previous phase were combined with a concatenate layer, then through a fully connected layer, the final prediction was made. The model was tested in both freeway and urban areas, and the results show that the performance is better in the urban area. The authors compared the model with ARIMA, SAE, LSTM, SVM, and CNN-LSTM, and the model outperformed all those models. Meanwhile, according to the results, the addition of BRNN to the model can improve the accuracy of the results (0.8 higher in MAE, 2.6% higher in MAPE, and 1.7 higher in RMSE).

Besides SRNN and BRNN, the Sequence to Sequence (seq2seq) structure is also an option for building up a Deep-Learning structure for RNN. The idea of seq2seq learning has been widely used in machine translation and claimed to have perfect performance [79,91]. The structure of seq2seq is shown in Fig. 11. As we can see, the model can be divided into two main parts, i.e., Encoder and Decoder. In the Encoder phase, the Encoder will encode the input source sequence into a vector $C = [h_1, \ldots, h_t]$ by using the hidden states h from the RNN modules inside the Encoder in each timestamp, which can be calculated as follows:

$$h_i = R(x_i, h_{i-1}), \tag{23}$$

where R can represent a single layer RNN or RNN in Deep-Learning structure. The vector C is believed to contains the high-level features in the input sequence. Then, the RNN modules inside the Decoder will not only take the previous timestamp's output and hidden state, but also the vector C to calculate its output. The reason for using the seq2seq structure is to output an indefinite length of the sequence, which is important in machine translation. Recent research found that it also performed well in time-series prediction tasks [19,92].

The application of seq2seq can be seen in [19]. The authors provided a model called Spatio-Temporal Attentive NN (STANN). The main structure of the model is based on a seq2seq structure, with the addition of a fully connected layer as an attention layer to the encoder and decoder. The attention layer in the encoder is used to extract the spatial features in the input data, and the attention layer in the decoder is used to extract the temporal features. The length of the input sequence was set at 12, which means the model took data from the past 120 min to make a prediction. The prediction intervals were set at 30, 60, 90, and 120-min, which can be seen as including both short-term predictions and long-term predictions. There were two-layers of LSTM in encoder and decoder. In the experiment, the authors compared the model with SVR, Random Forest (RF), seq2seq with no attention layer, STGCN, and DCRNN. The model showed better accuracy and scalability than the other models. Then, the authors analyzed the performance between the models, which has only one attention layer in either encoder and decoder, SANN and TANN respectively. The results showed that the two attention layers can be helpful in feature extraction. The authors pointed out in the experiment that with the increase of the attention layer, the training time and testing time were much greater than SANN and TANN, but are still shorter than STGCN and DCRNN.

The model's main structure in [92] is very much like the STANN model, in that they both have a seq2seq base model and both have two attention layers. However, there were still two main differences: First, in STANN, the spatial attention layer takes the historical traffic speed data directly without any preprocessing, but in this paper, the historical data will first be aggregated into an adjacency matrix; then, the attention layer will take the input as a graph but not a sequence. The second difference is the RNN used in the encoder and decoder is not longer than LSTM but Conv-GRU. As we reviewed before in this subsection, the structure of Conv-GRU is very like Conv-LSTM except that the gate structure of Conv-GRU is based on GRU, and the Conv-RNN can better capture the special features in the input dataset. The authors compared their model with SVR, RF, FFNN, SAE, GRU, LSTM, DCRNN, DNN-BTF, and evaluated by using MAE, RMSE, and WMAPE. The results show the model performs the best in 5, 15, 30, 60-min prediction intervals. The authors also showed that the model will perform better when the resolution of the input data is higher, which has more detailed features in the dataset.

One disadvantage of the seq2seq structure is that, although the vector C contains all high-level features in the input sequence when applying C into the decoder, C will have the same effect on the outputs for RNN in the different timestamp. This will waste the features extracted from the input while we face a multi-step prediction task.

The papers in the past ten years whose models used RNN as its main module are listed in Table 7. As we can see from the table, the number of research studies on using RNN for traffic prediction is larger than the other models we introduced before. The main reason is that the inner data processing structure of the RNN model can better explain the interaction from the data on different timestamps. Meanwhile, most of the prediction models these days try to use a DL structure, which further improves the accuracy of the prediction model. The RNN-based model can capture thanks to the Convolutional, seq2seq, and attention-based structures, the spatial–temporal correlation of the traffic network. However, researchers need to consider computation during training and real-world prediction tasks. Due to the repeat module structure of each RNN layer, it will cost much more time to train an RNN model, which is a severe problem to consider when deciding to use the RNN model in real applications. Meanwhile, high time consumption will also increase the difficulty of maintaining the prediction model, while the network's traffic patterns are changed.

3.3. Convolutional Neural Network (CNN)

CNN model used for traffic prediction has excellent features describing the spatial interactions among road segments within a big traffic network. The concept of the CNN was first introduced by Yann LeCun in the 1990s [98]. A CNN consists of at least one convolutional layer and a fully connected layer after the convolutional layer, and may also have at least one pooling layer. Due to the local link and the pooling process, the features learned from CNN include translational and rotational invariance. Through the methods of weight sharing and local perception filing, the number of weights is reduced, which significantly reduces the complexity of the network and improves the generalization ability of the network. CNN model has often been used in pattern recognition. In traffic prediction, some researchers use CNN to extract the spatial features between the related road sections.

The convolutional layer is obtained by moving the convolutional kernel according to a specified step size on the previous layer. The main purpose of convolution is to extract high-order features. A simple convolution process is shown in Fig. 12. If we assume the input data as $X \in \mathbb{R}^{M \times N}$, and $W \in \mathbb{R}^{I \times J}$ is the weights matrix for convolutional kernel, $b \in \mathbb{R}$, the length of the step is l, and $C \in \mathbb{R}^{P \times Q}$ is the output of the convolutional layer where $P = (M - I)/l + 1$ and $Q = (N - J)/l + 1$, we can get:

$$C_{p,q} = \sum_{i,j} W_{i,j} * X_{i+p*l-1, j+q*l-1} + b \tag{24}$$

Table 7
RNN based models in the past ten years.

Type	Papers	Task	Prediction interval	Area	I/O	Structure	Additions
LSTM	Luo et al. [20]	Flow	5-min	Freeway	Multi (36 detectors)-One	KNN (K = 10 for S339, K = 6 for S448) & Regression layer Single LSTM	Use KNN for extract spatial features. Classify the historical data by different workdays.
	Yi et al. [86]	Speed	5-min	Freeway	One–One	SRNN (LSTM)	GRU outperforms LSTM while the input length is shorter
	Yang et al. [93]	Flow	15-min	Freeway	One–One	Attention layer & SRNN (LSTM) & Regression layer	Use Attention layer to deal with super long (29 steps) input
	He et al. [19]	Speed	30, 60, 90, 120-min	Urban	Multi–Multi	Seq2seq (LSTM) with attention layers in encoder & decoder	The first Attention layer is used to extract spatial correlations from the input data.
	Yao et al. [94]	Flow	1-day	Urban	Region-based road network	CNN (kernel size = $3 * 3$) & Single LSTM & Attention layers	Multi-task. Context features in the input
	Zou et al. [85]	Flow	1-h	Urban	One–One	SRNN (LSTM)	
	Tian et al. [95]	Flow	5, 15, 60-min	Freeway	One–One	SRNN (LSTM-M) (6 layers)	Using extra parameters to deal with data lost
	Wang et al. [89]	Flow	15, 20, 25, 30-min	Freeway	One–One	Embedding layer & BRNN (LSTM) & Mean pooling layer & Regression layer	The BRNN layer is associated in a stacked style (6 layers).
	Liu et al. [90]	Flow	5-min	Freeway & Urban	Multi–Multi	(1D-CNN & SRNN(LSTM) or BRNN (LSTM)) & FFNN	1D-CNN preprocess the input data for SRNN at each time-step one at a time
	Dai et al. [96]	Flow	5-min	Freeway	One–One	Single layer FFNN (128 units) & single layer LSTM	Use FFNN to calculate the residual series. Pre-train all parts of the model separately
	Tian and Pan [78]	Flow	15, 30, 45, 60-min	Freeway	One–One	Single layer LSTM	
GRU	Zhao et al. [68]	Speed	15, 30, 45 , 60-min	Freeway	Multi–Multi	GCN & Single layer GRU	Use GCN to extract the spatial features. The effect of the number of units in GRU was studied.
	Guo et al. [18]	Flow	5 30-min	Freeway & Urban	One–One	Single layer GRU	Online multi-step model
ConvRNN	Yu et al. [97]	Speed	15, 30, 60-min	Freeway	Multi–Multi	Pooling layer & Seq2seq (Conv-GRU) & Un-Pooling layer	Use U-Net to implement Seq2seq
	Do et al. [22]	Speed	5, 15, 30, 60-min	Urban	Multi–Multi	Seq2seq (Conv-GRU) with attention layers in encoder and decoder	

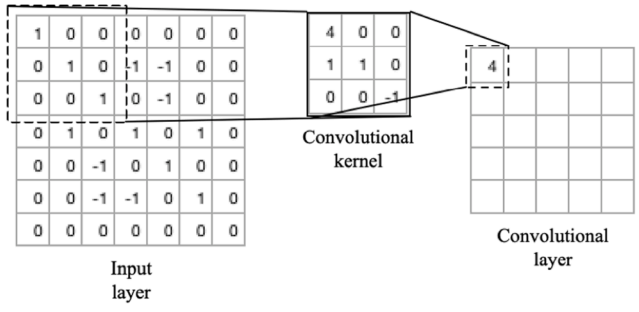

Fig. 12. An example of a simple convolution process for which the size of the input layer is 7*7, the size of convolutional kernel is 3*3, the size of moving step is 1, and the size of convolutional layer is 5*5.

The aim of having a pooling layer is to reduce the dimension of the feature from the convolutional layer, which can reduce the number of parameters and avoid overfitting. There are two common pooling methods: max pooling and mean pooling. In max pooling, the sample value will be the maximum value in the pooling window, and for mean pooling, the sample value will be the average value overall number in the pooling window.

The structure of CNN can help us capture the spatial features in the dataset. However, there is a drawback to using traditional CNN in traffic prediction tasks; when traffic records are input into the CNN, we must integrate the data into a graph-structure. However, the traditional CNN can only fit for the grid-structure. To overcome this problem, Defferrard et al. [99] provided a Graph-Convolutional NN (GCN) by introducing the Laplacian matrix into Convolutional NN. In [100], the authors used the GCN in their prediction model to mine the spatial correlation in a graph-structured dataset. The authors provided an ST-Conv block, which contains two gated-Conv layers for

temporal correlation extraction and a 3D GCN layer in between for spatial correlation extraction. The model was tested on three big traffic networks, which have 12, 228, and 1026 road detectors respectively. The results showed that the model outperforms the other state of the art models in all datasets, and has a lower time consumption than Conv-GRU.

In [101], the authors added an Attention layer to the ST-Conv block to further extract the spatial–temporal features. Moreover, the authors split the model into three parts, with each part taking historical records from the current day, one day ago and one week ago, respectively. In the end, a fusion layer combined the output from each part by weights. According to the results of the experiment, adding an attention layer can be helpful for improving the accuracy of the prediction model.

The above two papers used GCN to aggregate the traffic records by using the adjacency matrix, according to the spatial correlation. The problem is that the spatial correlation is influenced by the time changes in one day. To solve this problem, in [102], authors provided a 3D-GCN model by considering the time series' distance into the generation of the adjacency matrix. According to the result of the experiment on a big road network, the new model outperforms other state-out-the-art models.

The CNN-based model used in the past ten years are shown in Table 8. As we can see, the model is widely used for traffic prediction in a big road network. Meanwhile, according to past research, the GCN is more qualified than standard CNN when dealing with traffic prediction in a large transportation system. As we have addressed before, the main reason is that the spatial connection among road segments is not the same as the connection among pixels in a 2D picture. The spatial relationship of a transportation system is represented by the graph, i.e., adjacency matrix, where GCN is more suitable than standard CNN. The limitation of CNN models is that the spatial correlations extracted by the model only based on geographical information. The spatial correlations may change during different periods in one day or weekdays and weekends. Researchers should modify the structure of the model to suit the temporal change of spatial correlations, such as changing the adjacency matrix into the temporal adjacency matrix in [102].

4. Hybrid model

In this section, we will review the hybrid models used for traffic prediction. The results from multiple models are integrated by a combination method to make final prediction. Compared to normal ML model, a hybrid model can capture complicated relationships by using a relatively shallow model structure which has less computation consumption. The difficulty for using a hybrid model lies in finding an appropriate combination of methods to balance the influence of each models' results to the final prediction.

The hybrid model can be further separated into two small classes. Sub-models in the first class can make the prediction individually; the aim of the combination layer is to choose the best results, or choose the main results from one model, while the rest of the results are used to adjust the main results. In the second subset each sub-model is used to focus on one specific dataset feature. Meanwhile, each sub-model runs in a parallel way: the combination layer is used to combine the extracted feature, then it can make the final prediction [103].

We will first take a look at the first type. In [104], the authors chose NN as the aggregation layer for the three separate models, i.e., Moving Average (MA), Exponential Smoothing (ES), Autoregressive MA (ARIMA). The sub-models were given historical data collected in different periods, i.e., weekly, daily, and hourly. The NN is one hidden layer FFNN with a single output. The model performs very well at different prediction intervals. Meanwhile, the authors also pointed out that exchanging ARIMA with SARIMA does not improve the accuracy of the model.

In addition to using NN, the authors in [105] provided a more intuitive way to choose the prediction result between the sub-models. The authors combined a Historical Average method (HA) with assembled FFNNs. To assemble the FFNNs, the authors chose the Bagging method to dynamically distribute the training dataset for each FFNN to generalize the assembled system. Then, an Adaboosting method was used to decide the weights for each prediction result of the FFNNs to the final prediction result of the assembled system. In the end, the final prediction results will be given by the sub-model, which was chosen concerning the error in predicting the current traffic flow. If the error is beyond the threshold, assembled FFNN is chosen; otherwise, HA. According to the experiment results, the Bagging method can highly improve the prediction results, and the hybrid model outperforms the model with only the assembled FFNN.

There are also some examples of the second type of hybrid model. In [106], an attention-based hybrid model was proposed. The model starts with an attention layer, to determine how much weights the flow from different correlated positions on the previous time step will affect the prediction result. The sub-models will take the data from both a day before and one week before, in addition to the output from the attention layer. The final prediction will be carried out by using a regression layer. The model was tested on a dataset collected by a set of detectors on one freeway line. The model was used to predict traffic flow at different prediction intervals, and it outperformed the other state-of-the-art models. The authors further proved that by using the Attention layer, the model can better capture the useful correlations among the dataset.

The hybrid model has been used to capture seasonal features from different record periods or spatial–temporal features separately, as well as dealing with the prediction task in different traffic situations. In [107], the authors provided a hybrid method combined with GRU, ARIMA, and RBFNN to deal with the predictions in different traffic situations. The sub-models were chosen based on the Improved Bayesian Combination Method concerning the error rate of the prediction on the current time step. Meanwhile, the authors added a correlation analysis to select the most related historical data to input to the hybrid model to reduce the influence from the length of the input window. According to the results, the model was tested in different traffic scenarios and different prediction intervals, and showed better robustness, scalability, and accuracy.

Table 9 shows the hybrid model used for traffic prediction in the past ten years. The best feature of the hybrid model is that the model can take care of different dataset features by combining different types of models. For example, sub-models in [104] are used for predicting traffic flows containing different amounts of random vibrations; while in [106], sub-models are used to extract spatial and temporal features, respectively. Meanwhile, the combination method is also an essential part of the hybrid model. Some combination methods are used to select the most appropriate prediction result among the outputs of the sub-models [105]. In contrast, the other combination methods are used to decide how many efforts should output from one sub-models contributes to the final prediction. Despite the benefits of using a hybrid model, the user should consider the combination method and each sub-model's role to avoid redundant computation and unnecessary time costs.

5. Summary

In this section, we will give you an overall comparison of the ML model types we have discussed. As we can see from Table 10, we compare the models in the following aspects:

- Difficulty of implementation: How difficult to implement the model, including figuring out the structure of the model, the parameters or the hyper-parameters of the model. The accuracy of Regression, example-based and Kernel-based model is heavily dependent on the parameters, it will cost significant effort to find an adequate set of parameters.

Table 8
CNN-based models used for traffic prediction in the past 10 years.

Papers	Task	I/O	Structure	Additions
Yu et al. [100]	Speed	Multi–Multi	Two ST-Conv Block (= 1 Gated-Conv + GCN + Gated-Conv)	GCN is in a 3D structure, Kernel function is Chebyshev Polynomials Approximation.
Guo et al. [101]	Speed	Multi–Multi	3 ∗ Attention based ST-Conv (= Attention + GCN) + Fusion layer	Use Fusion layer for concatenate features from different sets of historical records.
Yu et al. [102]	Speed	Multi–Multi	3D-GCN with GLU activate function	Using Temporal adjacency matrix; Train the model based on both MAE and MSE

Table 9
Hybrid models used for traffic prediction in the past 10 years.

Papers	Task	Sub-models	Combination method	Additions
Tan et al. [104]	Flow	Moving average, exponential smoothing, ARIMA	NN	Separated model used for historical records collected in different periods.
Moretti et al. [105]	Flow	A-FFNN historical average (HA)	Based on the error on the prediction result for the current time step. HA, if smaller than the threshold, otherwise A-FFNN	Three FFNNs were assembled by using the Bagging algorithm and Adaboosting method.
Wu et al. [106]	Flow	CNN, GRU	Regression layer	Use attention layer in the beginning to extract the spatial–temporal correlation for the sub-models
Gu et al. [107]	Flow	GRU, ARIMA, RBFNN	Improved Bayesian Combination method	Correlation analysis for choosing records for prediction.

- Cost of implementation: It takes the training time and hardware dependency of the model into consideration. Normally, the NN model spends more time on model training, and the computing capacity of the hardware will affect the performance of the model.
- Dataset requirement: Does the model need a big dataset to guarantee the performance of the model? The NN model always needs a big dataset to adjust the parameters inside the model in a supervised way. Meanwhile, the Example-based model, i.e., KNN model, also has a relatively high requirement on the dataset, in order to build up a search pool of reliable candidates.
- Deep-learning structure: Determine if this type of model has Deep-learning structure.
- Ability of spatial feature and temporal feature extraction: Here, we mean whether the model has a clear structure to extract a specific feature.
- Time cost of prediction: How much time will the model cost to make the final prediction. Normally, this is highly related to the structure of the model. Greater complexity in the structure will result in greater time costs.
- Maintain cost: Includes the updates of the hyper-parameters of the model; for KNN it also includes the updates of the candidates search pool.
- Robustness: How well the model will deal with the changes of the length of the prediction interval, data loss, etc. Meanwhile, this also includes the performance when the model is applied to a bigger road network.

6. Open source datasets

The need for sufficient data is very important for implementing an ML model. The data will be used to train the ML model to make the model converge. A good dataset must provide sufficient types of traffic pattern information according to different traffic prediction tasks. The basic information is the traffic flow, traffic speed information, or both. When studying the spatial–temporal relationship, the dataset should have the geographical information of each data collection, and the traffic direction and connection between the collectors. In some cases, when studying the influence of holidays, special events, or weather changes, the dataset should have enough related information either. The other thing we must pay attention to is the aggregation interval of the dataset. A smaller aggregation interval will have more random and non-linear features, while for a longer aggregation dataset, the trend of the time-series is smoother, which will highly affect the structure of the prediction model.

Generally speaking, a dataset that has been used a lot is more credible than others, and has more experiment results to compare with. There are some open-source datasets used for traffic predictions, and a short introduction about each dataset will also be proposed.

6.1. Datasets collected from real-word

First, we will focus on the datasets collected from the real-word. These datasets contain traffic data collected by on-road loop detectors, toll gates, or other on-road/road-side devices. These datasets are then published by the government or organizations for research use.

- PeMS [108]: PeMS collected real-time data from more than 39,000 individual detectors on freeway network in California. PeMS provides the information including flow, occupancy, speed and other useful data of each lane for each detector point. The minimum interval of the data is 5 min, which is very suitable for short-term prediction. The data is collected from mainline, on-ramp, off-ramp, and from freeways to freeways intersections; the position of the detector and the length of each road section

Table 10
The overview comparison of different ML model type.

Model	Regression	Example-based	Kernel-based	NN			Hybrid
				FFNN	RNN	CNN	
Implementation difficulty	High	Middle	Middle	Low	Middle	Middle	Middle
Cost of implementation	Low	Low	Low	Middle	High	High	High
Dataset requirement	Low	Middle	Low	High	High	High	High
Deep-learning structure	No	No	No	Yes	Yes	Yes	Yes
Spatial feature extraction	No	No	No	No	Yes	No	Yes
Temporal feature extraction	No	Yes	No	No	No	Yes	Yes
Prediction time cost	Low	Middle	Low	Middle	High	Middle	High
Maintain cost	Low	High	Low	High	High	High	High
Robustness	Low	Middle	Middle	Middle	High	High	High

helps us to better understand the spatial relationship between different detectors. PeMS automatically fills the lost data using the historical average method. In the prediction task for a big road network, some subsection of PeMS such as PeMS (M/L), PeMS-Bay, PeMS-D4/D8, etc., can always be used as the datasets.

- CityPlus Smart City datasets [109]: The traffic data is collected from Denmark both in urban and rural areas in 2014 from February to November, with the exception of July. There are records about average speed, and the number of vehicles every 5 min. In addition, the dataset includes the geographical information of each detector, and the dataset also provides pollution, weather, and special events information, which is very useful for creating an ML model that takes hybrid information as its input.
- RTMC [110]: The traffic flow and occupancy information were collected every 30 s and aggregated into 5 min. The dataset provides data from 2011 to 2017. Meanwhile, the dataset also contains the location information in the dataset.

For more open-source real-word datasets can be found in [111].

6.2. Simulators for traffic network

A real-word dataset is essential for training a model. However, the existing real-word datasets may not correspond with the needs of the experiment. Such as the scale of the traffic network of the given real-word dataset is not large enough, or maybe we need more drive behaviors that are not provided by the real-word dataset. To solve these problems, a traffic simulation system is required to enrich the traffic dataset.

The traffic simulation system simulates a realistic traffic system by establishing a calculation model. The simulation models can be divided into macroscopic and microscopic simulation models [112].

In macroscopic traffic simulation, traffic flow is considered continuous compressible fluid, composed of many vehicles [113]. It focuses on the volume, density, and speed of the traffic flow and ignores the interactions between the vehicles inside the traffic flow. So, the macroscopic traffic simulators require fewer computer resources, and the simulation speed is faster. Here are some famous macroscopic traffic simulators are: TransCAD [114], EMME [115].

As we said, the interactions inside of the traffic flow are not described in macroscopic traffic simulators. On the contrary, microscopic traffic simulators focus on the location and speed of every vehicle inside the traffic flow [116]. The system simulates the traffic flow by modeling the behavior of each vehicle. In a microscopic traffic simulation system, the user can define the car following behavior, lane change behavior, destination information, etc. These features give the user opportunity to modeling a more elaborate and real traffic environment. Here are some famous microscopic traffic simulators: SUMO [117], VISSIM [118], SimTraffic [119], CORSIM [120], Paramics [121], MITSimLab [122], TransModeler [123].

7. Useful add-ons

The ML model has great features for capturing complicate correlations, dealing with the non-linear vibration in time-series, and building up a complicated model structure. In this section, we will discuss situations that arise in traffic prediction while using the ML model. Meanwhile, we will review and compare the existing solutions for these problems.

7.1. Data preprocessing

Data preprocessing is a very important step before feeding into the model. Data preprocessing will highlight the specific features we want the model to learn, help the model converge faster, avoid the model to be affected by the useless information, etc. However, there are still difficulties in choosing a suitable preprocessing method; the chosen method should cooperate with the model and the task itself. In An unsuitable preprocessing method, for example, the preprocessing is not sufficient; too much noise and too many outliers will mislead the model in the wrong direction. On the other hand, if the preprocessing cut off too many vibration features in the dataset, the model will not be able to fit the time-series.

The first and most important type of data preprocessing method, especially for regression models and NN models, is the Normalization method. One of the reasons we need a Normalization method is the activation function we use in the NN model. The output of the activation function usually belongs to a certain range; for example, the output range of Sigmoid function is from 0 to 1, the output range of Tanh function is from −1 to 1. If the dataset exceeds this range, it will highly reduce the meaning of using the activation function, resulting in a model that is harder to converge. The other reason to use a Normalization method is to make sure the ranges of different types of data are at the same level. For example, if the input contains both flow and speed information, the highest speed on a road will not exceed 200 in a normal case; however, the highest value of traffic flow can easily exceed 300, sometimes even 1000. This will also drag the model convergence speed.

The second type of preprocessing method is the denoising method. The denoising method aims to remove the outliers in the dataset and

make the time-series smoother. The most used method is the Moving Average method. In [93], the authors provided a denoising method that used historical average value to find outliers and used the historical average value on the previous timestamps to generate a new value for the outliers. In [60], the authors chose a median filter as their way of denoising the input data.

Another way of preprocessing data is by detrending. By using the detrending method, the original time-series will be decomposed into a seasonal trend and residual trend. The mostly frequently used detrending method is to subtract the historical average from the original time-series. In [53], the authors chose the Discrete Fourier Transform (DFT) as the detrending method. In [50,124], Empirical Model Decomposition (EMD) was used as the decomposition method. Compared to DFT, EMD can decompose the time-series into high-frequency to low-frequency trends. Higher frequencies contain datasets with more original features.

Finally, we address the method of dealing with missing data. A situation in which data is missing can be classified based on the length of the missing time interval, i.e., long-term missing and short-term random missing. The most commonly used method to impute the missing data is the historical average. A time interval that has no record will be replaced with the average value of the record at the same time of the day/week. Although the historical average method is easy to implement, it is a very rough method and sacrifices many high-level features. In [125], Probabilistic Principal Component Analysis (PPCA) is used to deal with the missing data.

7.2. Model design

A well-designed model should satisfy the aim of the task, such as a multi-step prediction, a multi-task prediction, a prediction for a road network, etc. The difficulty here is how to get more accurate prediction results without introducing more time consumption; in the meantime, the model can make use of the correlation from different types of input or the spatial–temporal correlation.

First, we focus on the multi-step and multi-task predictions. One famous method is to add a Multi-task layer at the end of the model [17,94]. In [18], the authors solved the multi-step prediction task by taking the last prediction results as the new record, adding them into the end of the input sequence for the next step's prediction.

As for the spatial–temporal relationship extraction, KNN is used to find the most correlated detectors over the dataset. The drawback of the KNN method is that when it is used there will be a huge time consumption if the number of detectors is very big, and the set of detectors can only be used for one specific point, which is less efficient. Therefore, some papers provided the model using traditional CNN combined with RNN [90,126]. As pointed out in [66], the original CNN cannot deal with the graph information. To use the graphed spatial features (adjacency matrix), [100,101] used GCN to replace the original CNN. Meanwhile, the addition of an Attention layer to extract the special-temporal features has also been used in many papers [19,92,101]. The drawback of using the adjacency matrix to extract spatial correlations is that the spatial relationship between different detectors can be different at different times of the day [127,128]. In [102], the authors provided a temporal adjacency matrix to overcome this problem.

After determining the basic structure of the model, it is very important to choose suitable hyper-parameters, such as that number of neighbors for a KNN model, parameters for the kernel function for an SVM model, number of layers and number of hidden units for a NN model, adequate learning rate, etc. One of the easiest ways is the grid search method [78,85]. The problem for the grid search method is that it will lead to great time consumption. There are some other ways to solve the problem, such as ALS [17], FFA [67], ABC [129], etc.

8. Open challenges

In this section, we will discuss the open challenges for traffic prediction by using the ML model. We categorize the challenges under two main topic: traffic prediction-related issues and model-related issues.

8.1. Traffic prediction-related issues

Under this topic, we will discuss the challenges caused by the features of the traffic patterns. Firstly, the short-term prediction task for a traffic network, which has traffic lights, is difficult. First of all, the traffic light on the road will cause short-term traffic condition changes, which will lead to significant flow fluctuation. Meanwhile, the traffic light will also cause the spatial relation change between two road segments. Besides, one city may change the way of controlling the traffic light. That will require the evolution of the traffic pattern learned by the model, which will increase the implementation cost of the prediction system.

Another challenge belongs to the prediction task in the urban-freeway region. The traffic patterns in the urban area are different on the freeway. The spatial relationship between these two regions is more complicated than in a single traffic environment. This led to the prediction system applied to the conjunction area of the urban road, and the freeway should have a clear distinction between these two different sets of traffic patterns. Meanwhile, the interactions between these two sets of traffic patterns should also be studied in the traffic prediction system.

Besides, the prediction scale of the system is also a challenge. Most state-of-art prediction systems focus on the macroscopic prediction of the traffic network, which helps congestion detection and route planning. However, different types of traffic participants have different on-road behaviors and diverse needs of prediction information. Meanwhile, the lanes within a road may have utterly different traffic features. For example, the speed limit may be changed, or the allowed vehicle type may differ on separate lanes in one road segment. The traffic prediction system should also consider these factors.

8.2. Model-related issues

There are also some model-related challenges. As we can see from the above review, the accuracy of the ML-based prediction models is very high. However, the final aim for developing a traffic prediction system is to use it in reality, e.g., provide real-time traffic information for ITS, or provide useful information to a vehicular cloud for the potential computing power calculation. To embed a traffic prediction into a complicated traffic system, there are some challenges.

The first challenge is from the update time interval of the real-word dataset. In general, there are two types of dataset, i.e., real-time and historical datasets. When we aggregate the traffic prediction system into a real-time system, the prediction interval should be relatively small, especially for a complicated road section. The dataset we use in the experiment environment guarantees that the predictor can use the data collected from the recent past; however, the dataset of a real-word implementation environment cannot guarantee a very short time interval update. Sometimes, we have to predict the next 5-min by using the historical data collected from 40-min ago, which will highly decrease the accuracy of the model.

Another challenge is to balance the cost of the computing hard device, the depth of the prediction model, and the accuracy of the model. One big restriction when using an ML-based model is the computing hard device. Nowadays, the ML-based models are always implemented on a GPU, which costs a lot to make sure the model can be trained and predict at a relatively high speed. However, to use the predictor in a real-life situation, it is very difficult to guarantee such a big computing center, especially for a big traffic road network; the cost will be extremely large if we keep the same standard for the hard

devices as in the lab. It is therefore very important to provide a model that can deal with a big traffic road network while keeping a small time consumption.

Finally, it is very important to provide an efficient way of updating the parameters for the prediction system. First, it is important to decide when to update the parameters. One drawback of ML-based prediction models is that the accuracy of the model is highly related to the training dataset. If the frequency to update the model is too low, the model will lose sensitivity for the changes in the traffic patterns of the given road section. On the contrary, if the frequency is too high, it will make the model lose the learned seasonal features of the given road section. Secondly, it is important to decide how to update the parameters. The method should keep the model sensitive to the new changes of the traffic patterns, but should also keep the seasonal features learned from the historical dataset. Meanwhile, the time consumption for the update method should be small.

9. Conclusion and future work

In this paper, we review the ML model in the field of traffic prediction, and also the current open challenges. As we can see, the model is complicated for mining high-level features. Also, we noticed that more and more papers have tried to use RNN and CNN or Conv-RNN together, and the structure of the seq2seq and Attention-based models are more and more popular. We also saw that the hybrid model has a lot of potential for digging high-level features, but keeps the structure of the model simpler and the time consumption lower. Meanwhile, the task for traffic prediction on a big road network has gained more and more attention recently.

In the future, we want to propose a model with high accuracy but maintains lower time consumption and a method that can model the spatial relation both on the geographical dimension and in a time dimension. A hybrid ML model combined with GCN and attention-based seq–seq model may be helpful for this task. GCN can extract the spatial relation on the geographical dimension based on the distance and connection information; Meanwhile, the spatial relationship on the time dimension can be retrieved by the attention layer based on the traffic flow information on each timestamp. The extracted features can be combined by using the appropriate combination method. Besides, we may also use KNN to separate the traffic network into small grids to reduce the computation cost.

CRediT authorship contribution statement

Azzedine Boukerche: Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing. **Jiahao Wang:** Conception and design of study, Acquisition of data, Analysis and/or interpretation of data, Writing - original draft, Writing - review & editing.

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.

Acknowledgments

This work is partially supported by NSERC-SPG, Canada, NSERC-DISCOVERY, Canada, Canada Research Chairs Program, NSERC-CREATE TRANSIT Funds, Canada.

All authors approved the version of the manuscript to be published.

References

[1] E. Dechenaux, S.D. Mago, L. Razzolini, Traffic congestion: an experimental study of the Downs-Thomson paradox, Exp. Econ. 17 (3) (2014) 461–487.
[2] W. Hu, Z. Feng, Z. Chen, J. Harkes, P. Pillai, M. Satyanarayanan, Live synthesis of vehicle-sourced data over 4G LTE, in: ACM MSWIM, 2017, pp. 161–170.
[3] W. Bao, D. Yuan, Z. Yang, S. Wang, B. Zhou, S. Adams, A. Zomaya, SFog: Seamless fog computing environment for mobile IoT applications, in: ACM MSWIM, 2018, pp. 127–136.
[4] D.L.L. Moura, A.L.L. Aquino, A.A.F. Loureiro, Towards data VSN offloading in VANETs integrated into the cellular network, in: ACM MSWIM, in: MSWIM '19, 2019, pp. 235–239.
[5] N. Nya, B. Baynat, Performance model for 4G/5G heterogeneous networks with different classes of users, in: ACM MSWiM, 2017, pp. 171–178.
[6] B. Olivieri, M. Endler, DADCA: An efficient distributed algorithm for aerial datacollection from wireless sensors networks by UAVs, in: ACM MSWiM, 2017, pp. 129–136.
[7] N. Aljeri, A. Boukerche, Movement prediction models for vehicular networks: An empirical analysis, Wirel. Netw. 25 (4) (2019) 1505–1518.
[8] N. Aljeri, A. Boukerche, A probabilistic neural network-based road side unit prediction scheme for autonomous driving, in: ICC 2019 - 2019 IEEE International Conference on Communications (ICC), 2019, pp. 1–6.
[9] B. Alipour, L. Tonetto, R. Ketabi, A. Yi Ding, J. Ott, A. Helmy, Where are you going next? A practical multi-dimensional look at mobility prediction, in: ACM MSWIM, 2019, pp. 5–12.
[10] A.C.S.A. Domingues, F.A. Silva, A.A.F. Loureiro, On the analysis of users' behavior based on mobile phone apps, in: ACM MobiWac, 2019, pp. 25–32.
[11] R.I. Meneguette, L.H. Nakamura, A flow control policy based on the class of applications of the vehicular networks, in: ACM MobiWac, 2017, pp. 137–144.
[12] T. Zhang, R.E.D. Grande, A. Boukerche, Vehicular cloud: Stochastic analysis of computing resources in a road segment, in: Proceedings of the ACM Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, & Ubiquitous Networks, 2015, pp. 9–16.
[13] P. Sun, N. AlJeri, A. Boukerche, A fast vehicular traffic flow prediction scheme based on fourier and wavelet analysis, in: IEEE Global Communications Conference, IEEE, 2018, pp. 1–6.
[14] V. Barnett, Environmental Statistics: Methods and Applications, John Wiley & Sons, 2005.
[15] M. Okawa, H. Kim, H. Toda, Online traffic flow prediction using convolved bilinear Poisson regression, in: IEEE International Conference on Mobile Data Management (MDM), IEEE, 2017, pp. 134–143.
[16] 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, Transp. Res. C 62 (2016) 21–34.
[17] W. Huang, G. Song, H. Hong, K. Xie, Deep architecture for traffic flow prediction: Deep belief networks with multitask learning, IET Intell. Transp. Syst. 15 (5) (2014) 2191–2201.
[18] J. Guo, Z. Wang, H. Chen, On-line multi-step prediction of short term traffic flow based on GRU neural network, in: Proceedings of the 2nd International Conference on Intelligent Information Processing, 2017, pp. 11:1–11:6.
[19] Z. He, C.-Y. Chow, J.-D. Zhang, STANN: A spatio–temporal attentive neural network for traffic prediction, IEEE Access 7 (2018) 4795–4806.
[20] X. Luo, D. Li, Y. Yang, S. Zhang, Spatiotemporal traffic flow prediction with KNN and LSTM, J. Adv. Transp. 2019 (2019).
[21] A.M. Nagy, V. Simon, Survey on traffic prediction in smart cities, Pervasive Mob. Comput. 50 (2018) 148–163.
[22] L.N. Do, N. Taherifar, H.L. Vu, Survey of neural network-based models for short-term traffic state prediction, Wiley Interdiscip. Rev.: Data Min. Knowl. Discovery 9 (1) (2019) e1285.
[23] M. Aqib, R. Mehmood, A. Alzahrani, I. Katib, A. Albeshri, S.M. Altowaijri, Smarter traffic prediction using big data, in-memory computing, deep learning and GPUs, Sensors 19 (9) (2019) 2206.
[24] A.B. Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. García, S. Gil-López, D. Molina, R. Benjamins, et al., Explainable artificial intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI, Inf. Fusion 58 (2020) 82–115.
[25] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[26] S. Shalev-Shwartz, S. Ben-David, Understanding Machine Learning, Cambridge University Press, 2014.
[27] J. Wang, A. Boukerche, The scalability analysis of machine learning based models in road traffic flow prediction, in: 2020 IEEE International Conference on Communications, IEEE, 2020, pp. 1–6.
[28] P. Sun, N. Aljeri, A. Boukerche, Machine learning-based models for real-time traffic flow prediction in vehicular networks, IEEE Netw. 34 (3) (2020) 178–185.
[29] T. Hastie, R. Tibshirani, J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Springer Science & Business Media, 2009.
[30] J. Rice, E. Van Zwet, A simple and effective method for predicting travel times on freeways, IEEE Trans. Intell. Transp. Syst. 5 (3) (2004) 200–207.

[31] X. Zhang, J.A. Rice, Short-term travel time prediction, Transp. Res. C 11 (3–4) (2003) 187–210.

[32] A.G. Hobeika, C.K. Kim, Traffic-flow-prediction systems based on upstream traffic, in: Proceedings of VNIS'94 - 1994 Vehicle Navigation and Information Systems Conference, IEEE, 1994, pp. 345–350.

[33] J. Kwon, B. Coifman, P. Bickel, Day-to-day travel-time trends and travel-time prediction from loop-detector data, Transp. Res. Rec.: J. Transp. Res. Board (1717) (2000) 120–129.

[34] X. Fei, C.-C. Lu, K. Liu, A bayesian dynamic linear model approach for real-time short-term freeway travel time prediction, Transp. Res. C 19 (6) (2011) 1306–1318.

[35] L. Li, X. Su, Y. Wang, Y. Lin, Z. Li, Y. Li, Robust causal dependence mining in big data network and its application to traffic flow predictions, Transp. Res. C 58 (2015) 292–307.

[36] H. Chang, Y. Lee, B. Yoon, S. Baek, Dynamic near-term traffic flow prediction: system-oriented approach based on past experiences, IET Intell. Transp. Syst. 6 (3) (2012) 292–305.

[37] L. Zhang, Q. Liu, W. Yang, N. Wei, D. Dong, An improved k-nearest neighbor model for short-term traffic flow prediction, Procedia-Soc. Behav. Sci. 96 (2013) 653–662.

[38] Z. Zheng, D. Su, Short-term traffic volume forecasting: A k-nearest neighbor approach enhanced by constrained linearly sewing principle component algorithm, Transp. Res. C 43 (2014) 143–157.

[39] 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.

[40] L.E. Peterson, K-nearest neighbor, Scholarpedia 4 (2) (2009) 1883.

[41] B.L. Smith, B.M. Williams, R.K. Oswald, Comparison of parametric and non-parametric models for traffic flow forecasting, Transp. Res. C 10 (4) (2002) 303–321.

[42] V. Vapnik, The Nature of Statistical Learning Theory, Springer science & business media, 2013.

[43] A. Ding, X. Zhao, L. Jiao, Traffic flow time series prediction based on statistics learning theory, in: Proceedings. the IEEE 5th International Conference on Intelligent Transportation Systems, 2002, pp. 727–730.

[44] L. Vanajakshi, L.R. Rilett, A comparison of the performance of artificial neural networks and support vector machines for the prediction of traffic speed, in: IEEE Intelligent Vehicles Symposium, 2004, 2004, pp. 194–199.

[45] Y. Zhang, Y. Xie, Forecasting of short-term freeway volume with v-support vector machines, Transp. Res. Rec. 2024 (1) (2007) 92–99.

[46] M. Castro-Neto, Y.-S. Jeong, M.-K. Jeong, L.D. Han, Online-SVR for short-term traffic flow prediction under typical and atypical traffic conditions, Expert Syst. Appl. 36 (3) (2009) 6164–6173.

[47] W.-C. Hong, Y. Dong, F. Zheng, C.-Y. Lai, Forecasting urban traffic flow by SVR with continuous ACO, Appl. Math. Model. 35 (3) (2011) 1282–1291.

[48] M. Lippi, M. Bertini, P. Frasconi, Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning, IEEE Trans. Intell. Transp. Syst. 14 (2) (2013) 871–882.

[49] M. Dorigo, V. Maniezzo, A. Colorni, Ant system: Optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. B 26 (1) (1996) 29–41.

[50] M. Duo, Y. Qi, G. Lina, E. Xu, A short-term traffic flow prediction model based on EMD and GPSO-SVM, in: IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), IEEE, 2017, pp. 2554–2558.

[51] J. Wang, Q. Shi, Short-term traffic speed forecasting hybrid model based on chaos–wavelet analysis-support vector machine theory, Transp. Res. C 27 (2013) 219–232.

[52] Z. Mingheng, Z. Yaobao, H. Ganglong, C. Gang, Accurate multisteps traffic flow prediction based on SVM, Math. Probl. Eng. 2013 (2013).

[53] X. Luo, D. Li, S. Zhang, Traffic flow prediction during the holidays based on DFT and SVR, J. Sensors 2019 (2019).

[54] S. Oh, Y.-J. Byon, K. Jang, H. Yeo, Short-term travel-time prediction on highway: a review of the data-driven approach, Transp. Rev. 35 (1) (2015) 4–32.

[55] F. Jabot, Why preferring parametric forecasting to nonparametric methods? J. Theoret. Biol. 372 (2015) 205–210.

[56] W.S. McCulloch, W. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (4) (1943) 115–133.

[57] F. Jin, S. Sun, Neural network multitask learning for traffic flow forecasting, in: IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), IEEE, 2008, pp. 1897–1901.

[58] B. Çetiner, M. Sari, O. Borat, A neural network based traffic-flow prediction model, Math. Comput. Appl. 15 (2) (2010) 269–278.

[59] K. Kumar, M. Parida, V. Katiyar, Short term traffic flow prediction for a non urban highway using artificial neural network, Procedia-Soc. Behav. Sci. 104 (2013) 755–764.

[60] N.G. Polson, V.O. Sokolov, Deep learning for short-term traffic flow prediction, Transp. Res. C 79 (2017) 1–17.

[61] Y. Lin, X. Dai, L. Li, F.-Y. Wang, Pattern sensitive prediction of traffic flow based on generative adversarial framework, IEEE Trans. Intell. Transp. Syst. 20 (6) (2018) 2395–2400.

[62] M. Arjovsky, S. Chintala, L. Bottou, Wasserstein generative adversarial networks, in: D. Precup, Y.W. Teh (Eds.), Proceedings of the 34th International Conference on Machine Learning, Vol. 70, PMLR, 2017, pp. 214–223.

[63] Y. Lv, Y. Duan, W. Kang, Z. Li, F.-Y. Wang, et al., Traffic flow prediction with big data: A deep learning approach, IEEE Trans. Intell. Transp. Syst. 16 (2) (2015) 865–873.

[64] L. Deng, D. Yu, et al., Deep learning: methods and applications, Found. Trends Signal Process. 7 (3–4) (2014) 197–387.

[65] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, A.Y. Ng, Multimodal deep learning, in: ICML, 2011, pp. 689–696.

[66] S. Goudarzi, M. Kama, M. Anisi, S. Soleymani, F. Doctor, Self-organizing traffic flow prediction with an optimized deep belief network for internet of vehicles, Sensors 18 (10) (2018) 3459.

[67] X.-S. Yang, Firefly algorithms for multimodal optimization, in: International Symposium on Stochastic Algorithms, Springer, 2009, pp. 169–178.

[68] L. Zhao, Y. Zhou, H. Lu, H. Fujita, Parallel computing method of deep belief networks and its application to traffic flow prediction, Knowl.-Based Syst. 163 (2019) 972–987.

[69] Y. Tao, P. Sun, A. Boukerche, A delay-based deep learning approach for urban traffic volume prediction, in: 2020 IEEE International Conference on Communications, ICC 2020, IEEE, 2020, pp. 1–6.

[70] M.I. Jordan, in: J. Diederich (Ed.), Artificial Neural Networks, 1990, pp. 112–127, chapter Attractor Dynamics and Parallelism in a Connectionist Sequential Machine.

[71] B.A. Pearlmutter, Learning state space trajectories in recurrent neural networks, Neural Comput. 1 (2) (1989) 263–269.

[72] A. Cleeremans, D. Servan-Schreiber, J.L. McClelland, Finite state automata and simple recurrent networks, Neural Comput. 1 (3) (1989) 372–381.

[73] S. Merity, Explaining and illustrating orthogonal initialization for recurrent neural networks, 2019, [Online] https://smerity.com/articles/2016/orthogonal_init.html. Accessed on 2019-10-01.

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

[75] F.A. Gers, J. Schmidhuber, F. Cummins, Learning to forget: continual prediction with LSTM, in: 1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470), Vol. 2, 1999, pp. 850–855.

[76] F.A. Gers, E. Schmidhuber, LSTM recurrent networks learn simple context-free and context-sensitive languages, IEEE Trans. Neural Netw. 12 (6) (2001) 1333–1340.

[77] F.A. Gers, N.N. Schraudolph, J. Schmidhuber, Learning precise timing with LSTM recurrent networks, J. Mach. Learn. Res. 3 (Aug) (2002) 115–143.

[78] Y. Tian, L. Pan, Predicting short-term traffic flow by long short-term memory recurrent neural network, in: IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), 2015, pp. 153–158.

[79] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1724–1734, http://dx.doi.org/10.3115/v1/D14-1179, URL: https://www.aclweb.org/anthology/D14-1179.

[80] R. Fu, Z. Zhang, L. Li, Using LSTM and GRU neural network methods for traffic flow prediction, in: 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), 2016, pp. 324–328.

[81] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W.-c. Woo, Convolutional LSTM network: A machine learning approach for precipitation nowcasting, in: Advances in Neural Information Processing Systems, 2015, pp. 802–810.

[82] R. Pascanu, C. Gulcehre, K. Cho, Y. Bengio, How to construct deep recurrent neural networks, in: Proceedings of the Second International Conference on Learning Representations (ICLR 2014), 2014.

[83] S. El Hihi, Y. Bengio, Hierarchical recurrent neural networks for long-term dependencies, in: Proceedings of the 8th International Conference on Neural Information Processing Systems, MIT Press, Cambridge, MA, USA, 1995, pp. 493–499.

[84] M. Hermans, B. Schrauwen, Training and analysing deep recurrent neural networks, in: Proceedings of the 8th International Conference on Neural Information Processing Systems, 2013, pp. 190–198.

[85] Z. Zou, P. Gao, C. Yao, City-level traffic flow prediction via LSTM networks, in: Proceedings of the 2nd International Conference on Advances in Image Processing, ACM, 2018, pp. 149–153.

[86] H. Yi, K.-H.N. Bui, H. Jung, Implementing a deep learning framework for short term traffic flow prediction, in: Proceedings of the 9th International Conference on Web Intelligence, Mining and Semantics, 2019, pp. 7:1–7:8.

[87] M. Schuster, K.K. Paliwal, Bidirectional recurrent neural networks, IEEE Trans. Signal Process. 45 (11) (1997) 2673–2681.

[88] Z. Cui, R. Ke, Y. Wang, Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction, 2018, CoRR abs/1801.02143. URL: http://arxiv.org/abs/1801.02143, arXiv:1801.02143.

[89] J. Wang, F. Hu, L. Li, Deep bi-directional long short-term memory model for short-term traffic flow prediction, in: Neural Information Processing, 2017, pp. 306–316.
[90] Y. Liu, H. Zheng, X. Feng, Z. Chen, Short-term traffic flow prediction with conv-LSTM, in: 2017 9th International Conference on Wireless Communications and Signal Processing (WCSP), 2017, pp. 1–6.
[91] I. Sutskever, O. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, in: Advances in Neural Information Processing Systems, 2014, pp. 3104–3112.
[92] L.N. Do, H.L. Vu, B.Q. Vo, Z. Liu, D. Phung, An effective spatial-temporal attention based neural network for traffic flow prediction, Transp. Res. C 108 (2019) 12–28.
[93] B. Yang, S. Sun, J. Li, X. Lin, Y. Tian, Traffic flow prediction using LSTM with feature enhancement, Neurocomputing 332 (2019) 320–327.
[94] H. Yao, X. Tang, H. Wei, G. Zheng, Z. Li, Revisiting spatial-temporal similarity: A deep learning framework for traffic prediction, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 5668–5675.
[95] Y. Tian, K. Zhang, J. Li, X. Lin, B. Yang, Lstm-based traffic flow prediction with missing data, Neurocomputing 318 (2018) 297–305.
[96] X. Dai, R. Fu, L. Yilun, L. Li, Deeptrend: a deep hierarchical neural network for traffic flow prediction, in: IEEE International Conference on Intelligent Transportation Systems (ITSC Workshop), 2017.
[97] B. Yu, H. Yin, Z. Zhu, St-unet: a spatio-temporal u-network for graph-structured time series modeling, 2019, arXiv preprint arXiv:1903.05631.
[98] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Comput. 1 (4) (1989) 541–551.
[99] M. Defferrard, X. Bresson, P. Vandergheynst, Convolutional neural networks on graphs with fast localized spectral filtering, in: Advances in Neural Information Processing Systems, 2016, pp. 3844–3852.
[100] B. Yu, H. Yin, Z. Zhu, Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting, in: J. Lang (Ed.), Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, ijcai.org, 2018, pp. 3634–3640.
[101] S. Guo, Y. Lin, N. Feng, C. Song, H. Wan, Attention based spatial-temporal graph convolutional networks for traffic flow forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 33, 2019, pp. 922–929.
[102] B. Yu, M. Li, J. Zhang, Z. Zhu, 3D graph convolutional networks with temporal graphs: A spatial information free framework for traffic forecasting, 2019, arXiv preprint arXiv:1903.00919.
[103] A. Boukerche, J. Wang, A performance modeling and analysis of a novel vehicular traffic flow prediction system using a hybrid machine learning-based model, Ad Hoc Netw. 106 (2020) 102224.
[104] M.-C. Tan, S.C. Wong, J.-M. Xu, Z.-R. Guan, P. Zhang, An aggregation approach to short-term traffic flow prediction, IEEE Trans. Intell. Transp. Syst. 10 (1) (2009) 60–69.
[105] F. Moretti, S. Pizzuti, S. Panzieri, M. Annunziato, Urban traffic flow forecasting through statistical and neural network bagging ensemble hybrid modeling, Neurocomputing 167 (2015) 3–7.
[106] Y. Wu, H. Tan, L. Qin, B. Ran, Z. Jiang, A hybrid deep learning based traffic flow prediction method and its understanding, Transp. Res. C 90 (2018) 166–180.
[107] Y. Gu, W. Lu, X. Xu, L. Qin, Z. Shao, H. Zhang, An improved Bayesian combination model for short-term traffic prediction with deep learning, IEEE Trans. Intell. Transp. Syst. (2019).
[108] Caltrans, Caltrans PeMS, 2019, http://pems.dot.ca.gov/. Accessed on 2019-10-01.
[109] M.I. Ali, F. Gao, A. Mileo, CityBench: A configurable benchmark to evaluate RSP engines using smart city datasets, in: In Proceedings of ISWC 2015 - 14th International Semantic Web Conference, W3C, Bethlehem, PA, USA, 2015, pp. 374–389.
[110] T.M. Kwon, TDRL, RTMC traffic dataset, 2019, http://www.d.umn.edu/~tkwon/TMCdata/TMCarchive.html/. Accessed on 2019-10-01.
[111] graohhopper, Open-traffic-collection, 2019, https://github.com/graphhopper/open-traffic-collection/. Accessed on 2019-10-01.
[112] K. Abdelgawad, S. Henning, P. Biemelt, S. Gausemeier, A. Trächtler, Advanced traffic simulation framework for networked driving simulators, IFAC PapersOnLine 49 (2016) 101–108.
[113] H.J. Payne, FREFLO: A macroscopic simulation model of freeway traffic, Transp. Res. Rec. (722) (1979).
[114] TransCAD, Transcad, 2019, https://www.caliper.com/tcovu.htm. Accessed on 2019-10-01.
[115] inrosoftware, EMME, 2019, https://www.inrosoftware.com/en/products/emme/. Accessed on 2019-10-01.
[116] M. Maciejewski, A comparison of microscopic traffic flow simulation systems for an urban area, Transp. Probl. 5 (2010) 27–38.
[117] SUMO, SUMO, 2019, https://sumo.dlr.de/index.html. Accessed on 2019-10-01.
[118] P. Vissim, VISSIM, 2019, https://www.ptvgroup.com/en/solutions/products/ptv-vissim/. Accessed on 2019-10-01.
[119] Trafficware, SimTraffc, 2019, https://www.trafficware.com/blog/category/simtraffic. Accessed on 2019-10-01.
[120] McTrans Center, U.o.F., TSIS-CORSIM, 2019, http://mctrans.ce.ufl.edu/featured/tsis/. Accessed on 2019-10-01.
[121] Ltd, P.M.S., Paramics, 2019, https://www.paramics.co.uk/en/. Accessed on 2019-10-01.
[122] ITSLAB, M., MITSIMLAB, 2019, https://its.mit.edu/software/mitsimlab. Accessed on 2019-10-01.
[123] Caliper, Transmodeler, 2019, https://www.caliper.com/transmodeler/default.htm. Accessed on 2019-10-01.
[124] J. Ke, H. Zheng, H. Yang, X.M. Chen, Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach, Transp. Res. C 85 (2017) 591–608.
[125] C. Chen, Y. Wang, L. Li, J. Hu, Z. Zhang, The retrieval of intra-day trend and its influence on traffic prediction, Transp. Res. C 22 (2012) 103–118.
[126] J. Zhang, Y. Zheng, D. Qi, Deep spatio-temporal residual networks for city-wide crowd flows prediction, in: Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[127] Y. Tao, P. Sun, A. Boukerche, A hybrid stacked traffic volume prediction approach for a sparse road network, in: 2019 IEEE Symposium on Computers and Communications, IEEE, 2019, pp. 1–6.
[128] Y. Tao, P. Sun, A. Boukerche, A novel travel-delay aware short-term vehicular traffic flow prediction scheme for VANET, in: IEEE Wireless Communications and Networking Conference, IEEE, 2019, pp. 1–6.
[129] D. Chen, Research on traffic flow prediction in the big data environment based on the improved RBF neural network, IEEE Trans. Ind. Inf. 13 (4) (2017) 2000–2008.

Azzedine Boukerche (FIEEE, FEiC, FCAE, FAAAS) is a Distinguished University Professor and holds a Canada Research Chair Tier-1 position with the University of Ottawa. He is founding director of the PARADISE Research Laboratory and the DIVA Strategic Research Center, and NSERC-CREATE TRANSIT at University of Ottawa. He has received the C. Gotlieb Computer Medal Award, Ontario Distinguished Researcher Award, Premier of Ontario Research Excellence Award, G. S. Glinski Award for Excellence in Research, IEEE Computer Society Golden Core Award, IEEE CS-Meritorious Award, IEEE TCPP Leaderships Award, IEEE ComSoc ComSoft and IEEE ComSoc ASHN Leaderships and Contribution Award, and University of Ottawa Award for Excellence in Research. He serves as an Editor-in-Chief for ACM ICPS and Associate Editor for several IEEE transactions and ACM journals, and is also a Steering Committee Chair for several IEEE and ACM international conferences. His current research interests include sustainable sensor networks, autonomous and connected vehicles, wireless networking and mobile computing, wireless multimedia, QoS service provisioning, performance evaluation and modeling of large-scale distributed and mobile systems, and large scale distributed and parallel discrete event simulation. He has published extensively in these areas and received several best research paper awards for his work. He is a Fellow of IEEE, a Fellow of the Engineering Institute of Canada, the Canadian Academy of Engineering, and the American Association for the Advancement of Science.

Jiahao Wang is currently a master student of computer science in Paradise Research Laboratory at the University of Ottawa. He received the B.Eng. degree in computer science from Sichuan University, Chengdu, Sichuan, China, in 2017. His current research interests includes traffic flow prediction, vehicular cloud.